これどうやって動いてるんだ？
Spring Framework/Bootのソースを読む

2022/02/10
Server-Side Kotlin Meetup vol.1

ユビー

About Me

Toshihiro Yagi  **@sys1yagi**

Working at  **ユビー**

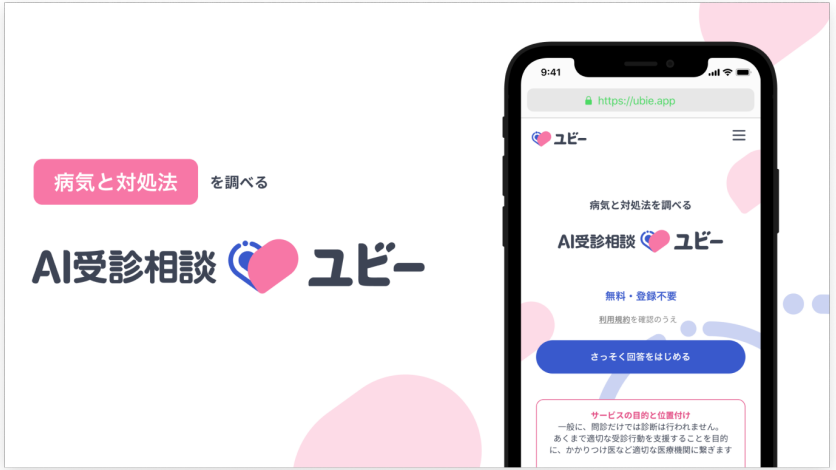
Software Engineer

Spring Boot/Kotlin, Rails/Ruby, React/TypeScript

Our Services

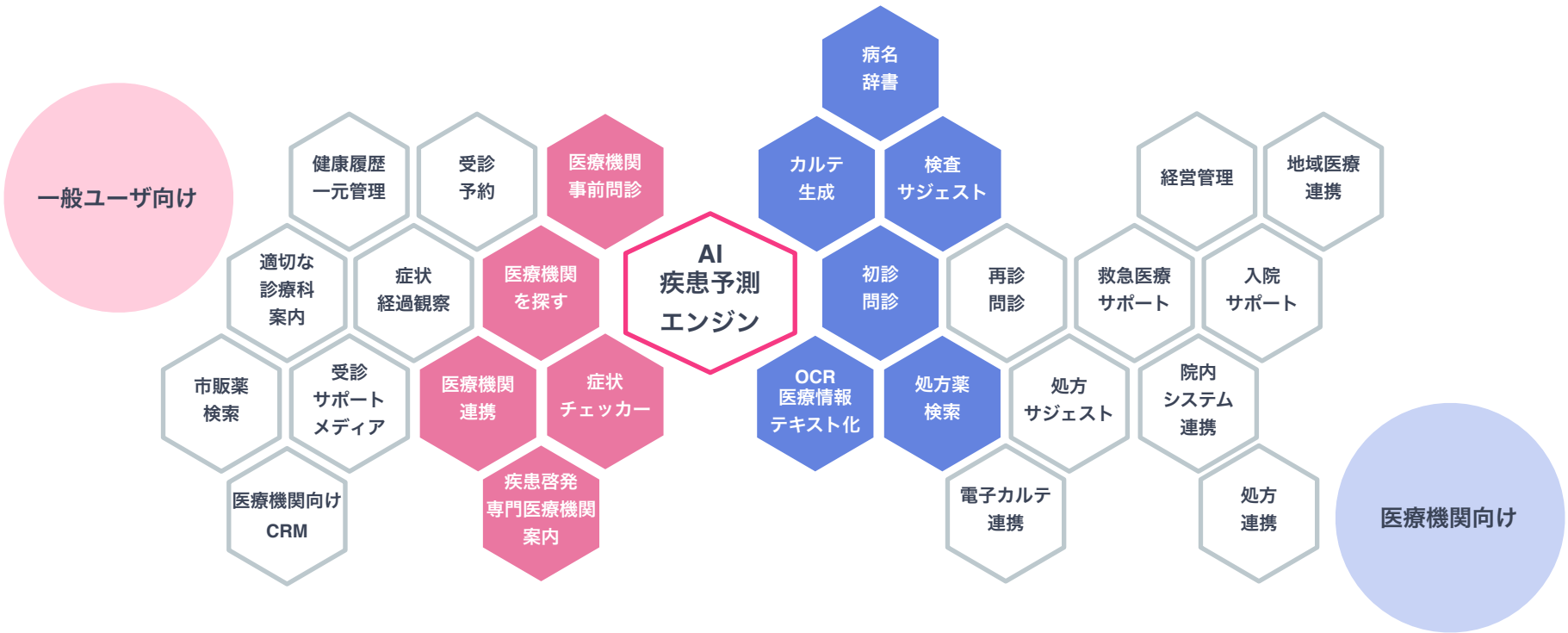
医療機関向け AI問診システム (toB)

生活者向け 受診相談アプリ (toC)



Our Services

AI疾患予測エンジンを中心に、toC、toB両面で今後も新たなプロダクトを同時に作っていく予定です



https://recruit.ubie.life/engineer

Spring Boot使っている方
どのくらいいますか？

Spring Frameworkも

Spring Bootも

かなりマジカル

ソースを読めば
だいたい分かる

今日話すこと

- Spring Framework, Spring Bootの概要
- Spring Framework, Spring Bootのソースのありか、読み進め方
- REST APIが動くまでを理解する

Spring Framework, Spring Bootの概要

Spring Framework

- 2004年に正式リリース。Javaアプリケーション用のフレームワーク
- Inversion of controlが中心的な機能
- AOP、データアクセスフレームワーク、トランザクション、MVC、リモートアクセス、バッチなどのモジュールがある
- 2018年9月 Spring Framework 5.1 から Kotlin 1.1サポート
    - https://spring.pleiades.io/guides/tutorials/spring-boot-kotlin/

https://en.wikipedia.org/wiki/Spring_Framework

Spring Boot

- 2014年に1.0が登場
- スタンドアロンのSpringアプリケーションを作成できる
- 各種構成を行うスターターを提供（Web、バッチ、メール、ログ、DBなど）
- AutoConfigurationによるサードパーティも含めた自動的な設定
- Spring Initializerは現在はSpring Bootを前提にしている
    - https://start.spring.io/

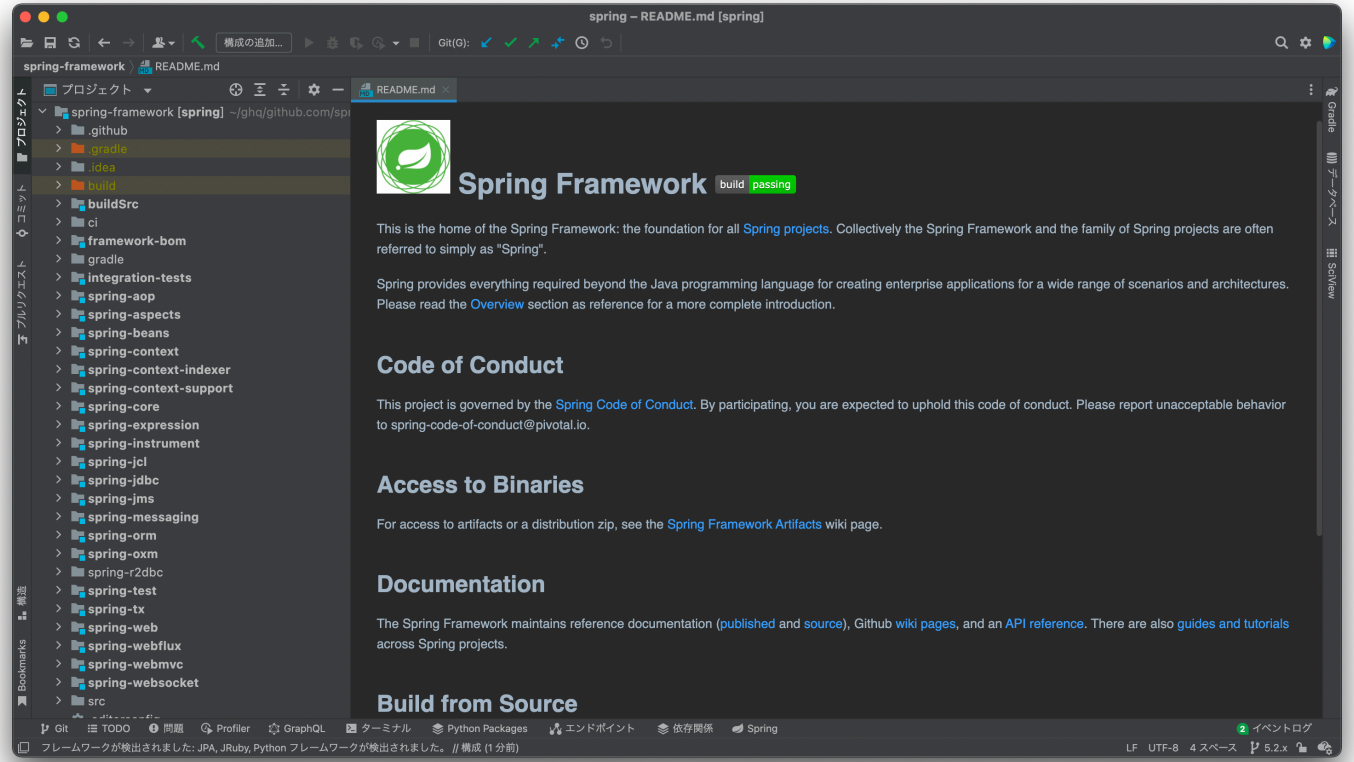
https://spring.io/projects/spring-boot

Spring Framework, Spring Bootの
ソースのありか、読み進め方

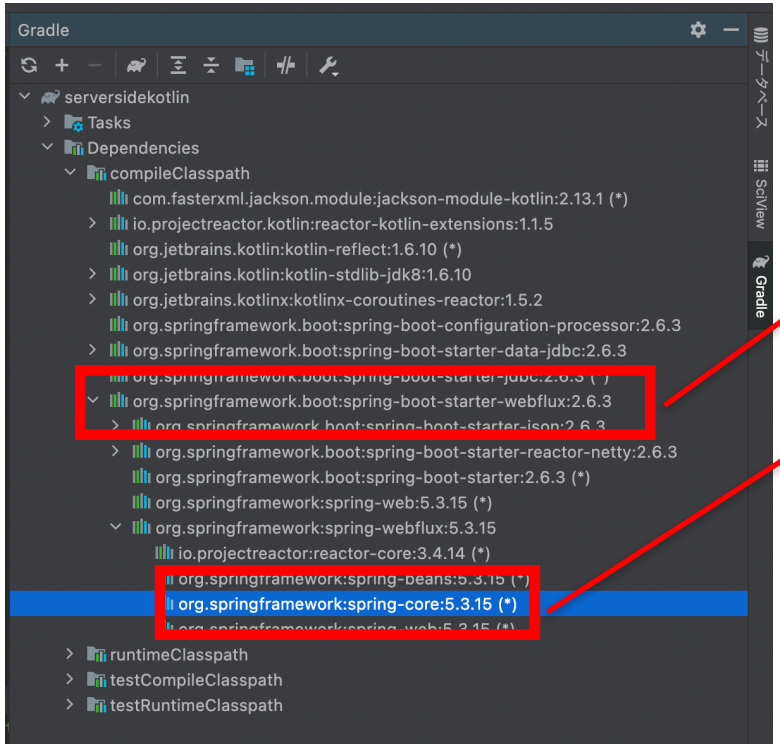
Spring FrameworkとBootのソースはGithubにある

- Spring Framework
  - https://github.com/spring-projects/spring-framework
  - git clone --depth 1 git@github.com:spring-projects/spring-framework.git
- Spring Boot
  - https://github.com/spring-projects/spring-boot
  - git clone --depth 1 git@github.com:spring-projects/spring-boot.git

IntelliJで普通に開けます



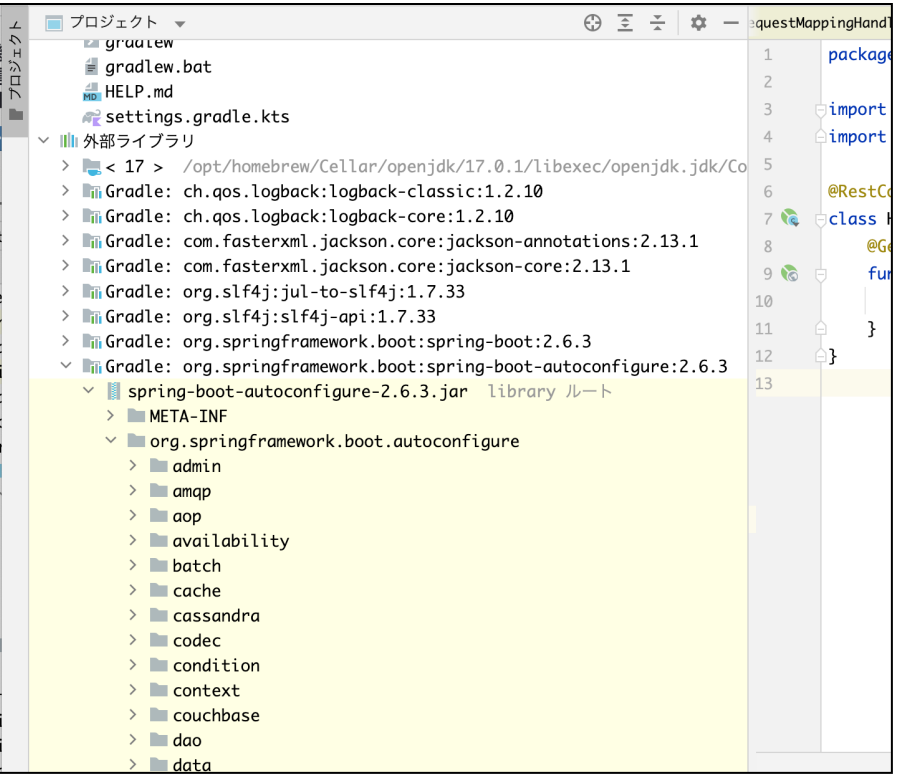
利用バージョンに合わせてブランチを切り替える



サイドバーのGradleで依存関係を見ると調べられる

- **Spring Boot 2.6.3**
  - git checkout -b 2.6.x origin/2.6.x
- **Spring Framework 5.3.15**
  - git checkout -b 5.3.x origin/5.3.x

Spring Bootアプリケーションのdependenciesを追う



- プロジェクトのツリー下部の「外部ライブラリ」から各dependenciesの詳細を見られる。大体sources.jarも付いているので実装を追いかけられる。
- 全体を眺めるときはSpring Framework, Bootを直接開くのが便利
- 特定の処理を追いかけるときはアプリケーションからブレークポイント張ってsources.jar内を追いかけるのが便利

REST APIが動くまで

Dependencies

- Spring InitializerでSpring Webだけ選択

```
implementation("org.springframework.boot:spring-boot-starter-web")
implementation("com.fasterxml.jackson.module:jackson-module-kotlin")
implementation("org.jetbrains.kotlin:kotlin-reflect")
implementation("org.jetbrains.kotlin:kotlin-stdlib-jdk8")
```

https://start.spring.io/

APIを実装する

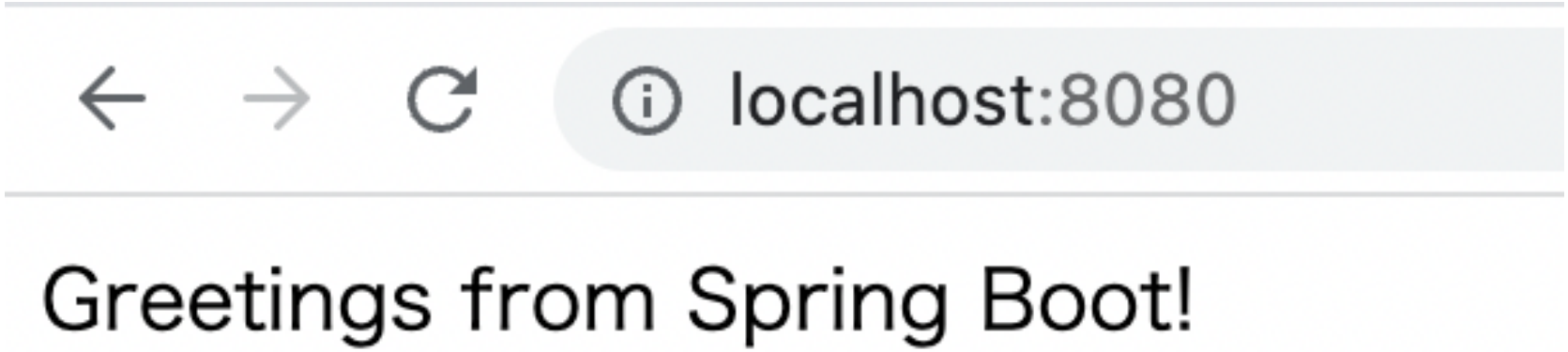
```
package com.example.controller

@RestController
class HelloController {
    @GetMapping("/")
    fun index(): String {
        return "Greetings from Spring Boot!"
    }
}
```

https://spring.io/guides/gs/spring-boot/

Spring Applicationのエントリポイント

```
package com.example

@SpringBootApplication
class ServersideKotlinApplication

fun main(args: Array<String>) {
    runApplication<ServersideKotlinApplication>(*args)
}
```

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.6.3)

2022-02-06 22:19:54.324  INFO 79807 --- [  restartedMain] c.e.s.ServersideKotlinApplicationKt      : Starting ServersideKotlinApplicationKt using Ja
2022-02-06 22:19:54.324  INFO 79807 --- [  restartedMain] c.e.s.ServersideKotlinApplicationKt      : No active profile set, falling back to default
2022-02-06 22:19:54.345  INFO 79807 --- [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spring.
2022-02-06 22:19:54.346  INFO 79807 --- [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider set
2022-02-06 22:19:54.754  INFO 79807 --- [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running on port 35729
2022-02-06 22:19:54.812  INFO 79807 --- [  restartedMain] o.s.b.web.embedded.netty.NettyWebServer  : Netty started on port 8080
2022-02-06 22:19:54.818  INFO 79807 --- [  restartedMain] c.e.s.ServersideKotlinApplicationKt      : Started ServersideKotlinApplicationKt in 0.62 s
```



SpringBootApplicationアノテーション

```
package com.example

@SpringBootApplication
class ServersideKotlinApplication

fun main(args: Array<String>) {
    runApplication<ServersideKotlinApplication>(*args)
}
```

SpringBootApplicationアノテーション

```
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
@Documented
@Inherited
@SpringBootConfiguration
@EnableAutoConfiguration
@ComponentScan(excludeFilters = { @Filter(type = FilterType.CUSTOM, classes = TypeExcludeFilter.class),
                    @Filter(type = FilterType.CUSTOM, classes = AutoConfigurationExcludeFilter.class) })
public @interface SpringBootApplication {
```

複数のアノテーションを合成している

AutoConfigurationを有効にする

DIのためのコンポーネントなどをスキャンする設定

Spring Boot

https://github.com/spring-projects/spring-boot/blob/main/spring-boot-project/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/SpringBootApplication.java

余談: 同じアノテーションをつけたら動く

```
@SpringBootConfiguration
@EnableAutoConfiguration
@ComponentScan(
    excludeFilters = [ComponentScan.Filter(
        type = FilterType.CUSTOM,
        classes = [TypeExcludeFilter::class]
    ), ComponentScan.Filter(type = FilterType.CUSTOM, classes = [AutoConfigurationExcludeFilter::class])]
)
class ServersideKotlinApplication
```

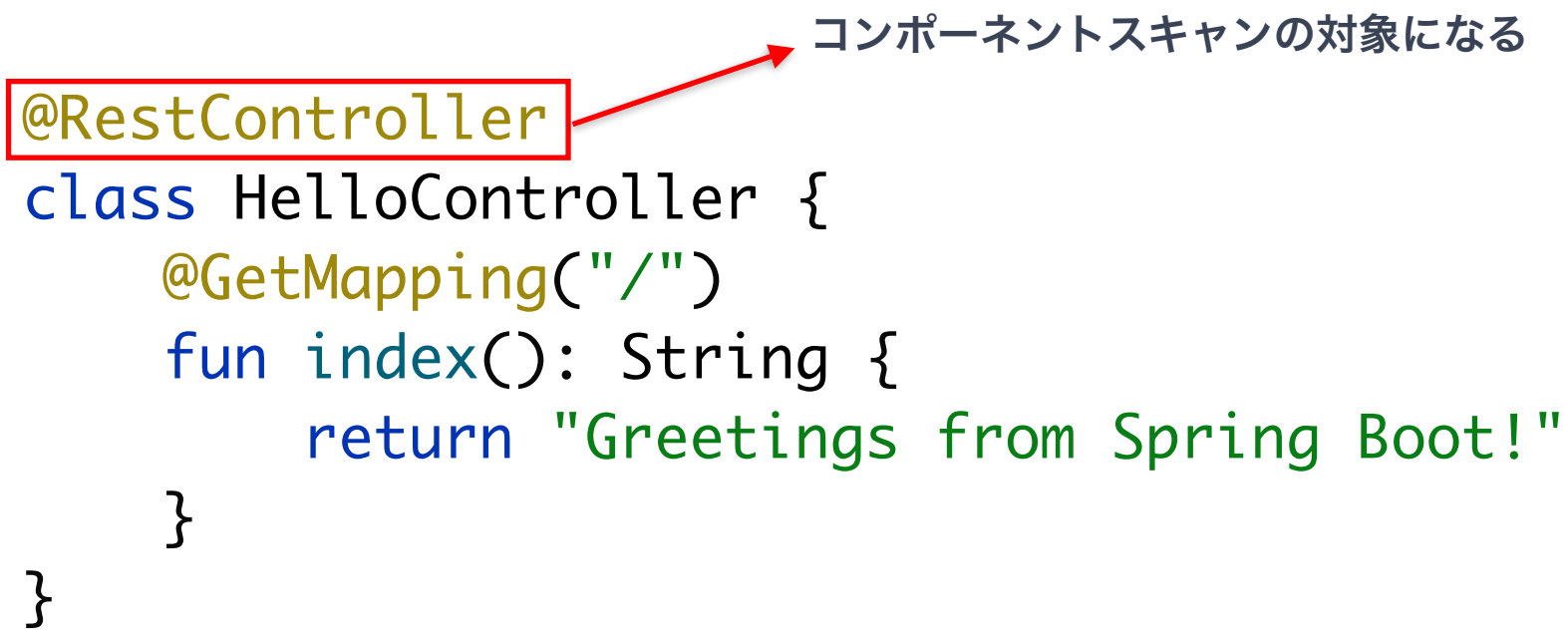
ConfigurationClassParserでコンポーネントをスキャンする

```
Set<AnnotationAttributes> componentScans =
  AnnotationConfigUtils.attributesForRepeatable(
    sourceClass.getMetadata(),
    ComponentScans.class,
    ComponentScan.class
  );
if (!componentScans.isEmpty()…
```

このアノテーションの付いたクラスのパッケージ配下を捜査し、インスタンスの生成や初期化処理を行う。SpringApplicationの中で行っている。

https://github.com/spring-projects/spring-framework/blob/main/spring-context/src/main/java/org/springframework/context/annotation/ConfigurationClassParser.java#L289

@RestControllerもComponentアノテーションを合成している



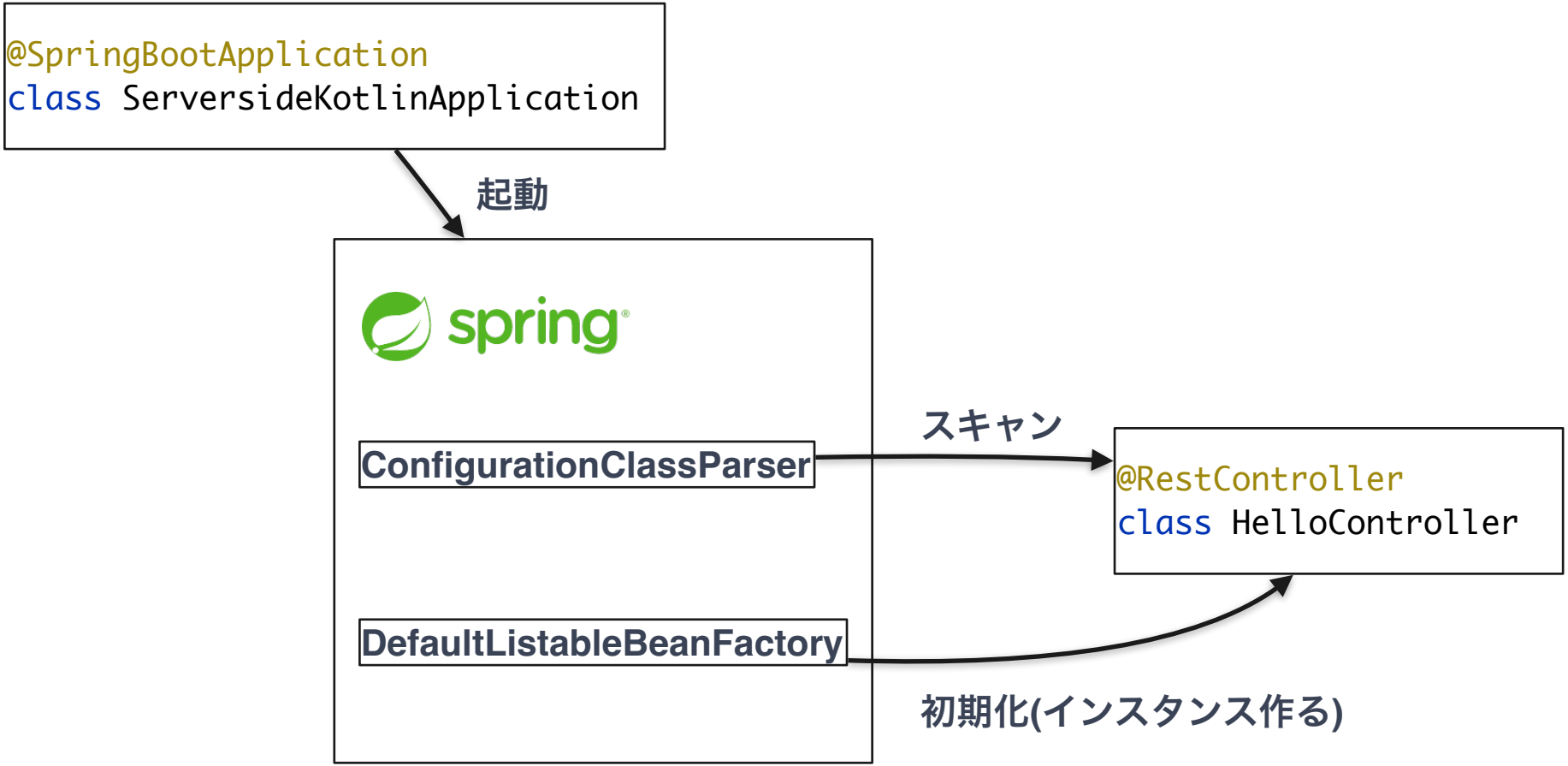
https://github.com/spring-projects/spring-framework/blob/main/spring-web/src/main/java/org/springframework/web/bind/annotation/RestController.java

DefaultListableBeanFactoryでインスタンスを作る

コンポーネントスキャンで集めた情報を使ってDefaultListableBeanFactoryでインスタンスを作る

```
for (String beanName : beanNames) {
  RootBeanDefinition bd = getMergedLocalBeanDefinition(beanName);
  if (!bd.isAbstract() && bd.isSingleton() && !bd.isLazyInit()) {
    if (isFactoryBean(beanName)) {
      // 省略
    }
    else {
      getBean(beanName);
    }
  }
}
```

https://github.com/spring-projects/spring-framework/blob/main/spring-beans/src/main/java/org/springframework/beans/factory/support/DefaultListableBeanFactory.java#L908


@SpringBootApplication
class ServersideKotlinApplication
起動
spring
ConfigurationClassParser
スキャン
@RestController
class HelloController
DefaultListableBeanFactory
初期化(インスタンス作る)

コントローラの関数とパスをどこかでマッピングしているはず

```
HelloController$index -> GET [/]
```

```
@RestController
class HelloController {
    @GetMapping("/")
    fun index(): String {
        return "Greetings from Spring Boot!"
    }
}
```

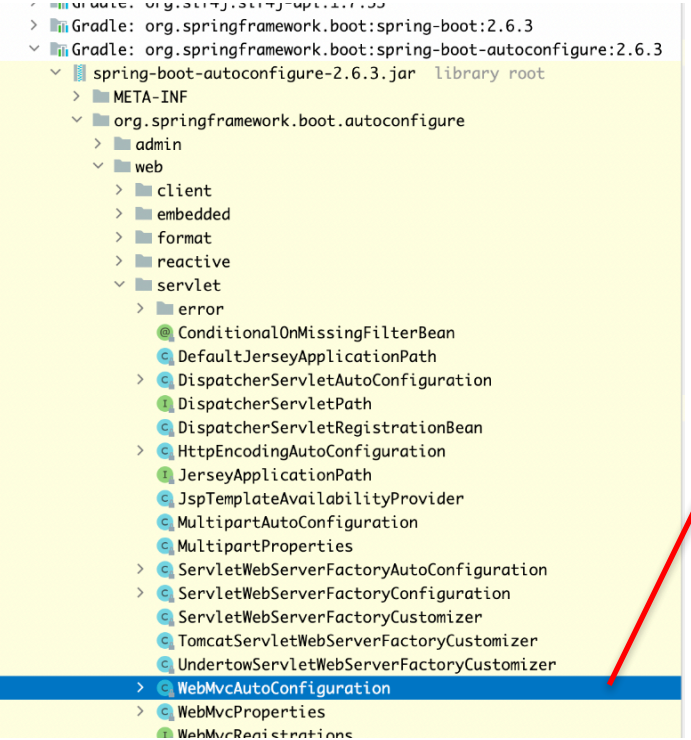
https://github.com/spring-projects/spring-framework/blob/main/spring-web/src/main/java/org/springframework/web/bind/annotation/GetMapping.java#L46

起動時はBoot、実行時はSpringかライブラリ

- 自動的な構成を提供するので基本的にアプリケーション起動時に処理が走る
- 関係しそうなAuto Configurationを頑張って探してブレークポイントを張ることになる

- アプリケーション起動後は基本的にSpringかライブラリの動作
- ブレークポイントを張れば大体追いかけられる
- AOPは動的なのでちょっとややこしい

Web系の機能なのでWebモジュールになんかありそう



- spring-boot-autoconfigureの中に、

  「WebMvcAutoConfiguration」というクラスがある

- spring-boot-starter-webが必要なモジュールを自動

  で追加している

  - spring-web
  - spring-webmvc
  - spring-boot-autoconfigure

WebMvcAutoConfigurationで怪しいBeanを探す

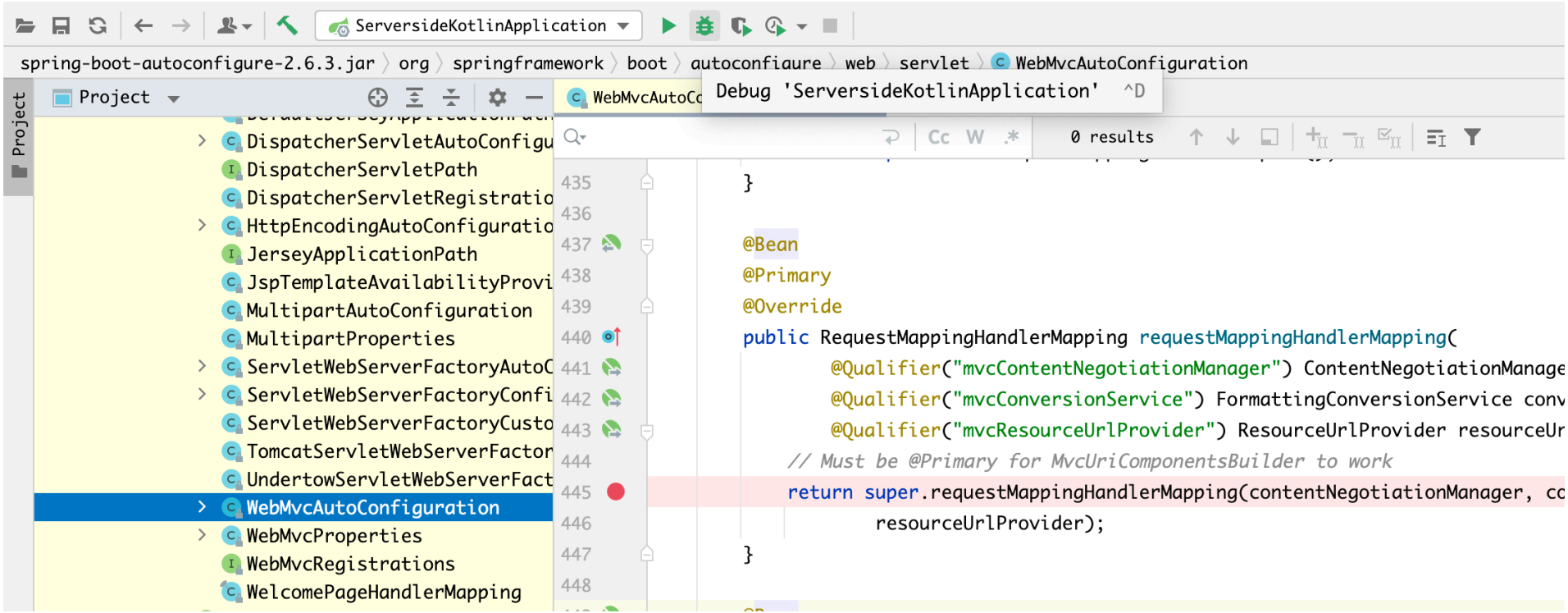
```
@Bean
@Primary
@Override
public RequestMappingHandlerMapping requestMappingHandlerMapping(
  @Qualifier("mvcContentNegotiationManager") ContentNegotiationManager contentNegotiationManager,
  @Qualifier("mvcConversionService") FormattingConversionService conversionService,
  @Qualifier("mvcResourceUrlProvider") ResourceUrlProvider resourceUrlProvider) {
    // Must be @Primary for MvcUriComponentsBuilder to work
    return super.requestMappingHandlerMapping(contentNegotiationManager, conversionService,
                                              resourceUrlProvider);
}
```

- GetMappingはRequestMappingを合成しているアノテーション
- RequestMappingHandlerMappingという名前で如何にも怪しい

Spring Boot

https://github.com/spring-projects/spring-boot/blob/main/spring-boot-project/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/web/servlet/WebMvcAutoConfiguration.java#L439

ブレークポイントを張って実行して調べる



RequestMappingHandlerMappingでリクエストパスと関数を紐つけている

```
public void afterPropertiesSet() {
  this.config = new RequestMappingInfo.BuilderConfiguration();
  this.config.setTrailingSlashMatch(useTrailingSlashMatch());
  this.config.setContentNegotiationManager(getContentNegotiationManager());
  // 略

  super.afterPropertiesSet();
}
```

Beanの初期化がすべて終わったあとに呼び出される関数

https://github.com/spring-projects/spring-framework/blob/main/spring-webmvc/src/main/java/org/springframework/web/servlet/mvc/method/annotation/RequestMappingHandlerMapping.java#L205

AbstractHandlerMethodMapping<T> (RequestMappingHandlerMappのスーパークラス)

```
@Override
public void afterPropertiesSet() {
  initHandlerMethods();
}
```

めっちゃそれっぽい

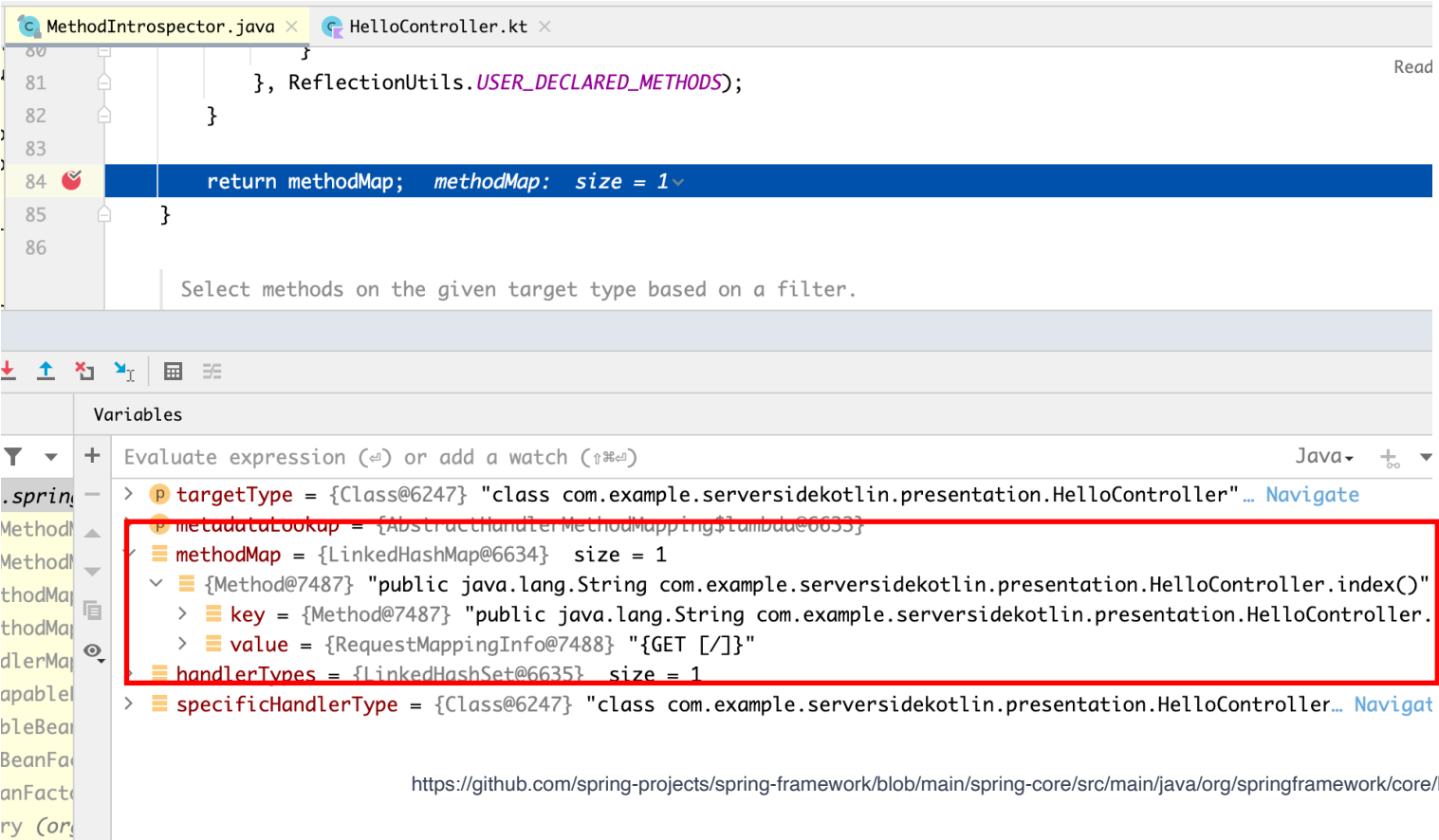
https://github.com/spring-projects/spring-framework/blob/main/spring-webmvc/src/main/java/org/springframework/web/servlet/handler/AbstractHandlerMethodMapping.java#L212

AbstractHandlerMethodMapping<T> (RequestMappingHandlerMappのスーパークラス)

```
protected void detectHandlerMethods(Object handler) {
  Class<?> handlerType = (handler instanceof String ?
      obtainApplicationContext().getType((String) handler) : handler.getClass());

  if (handlerType != null) {
    Class<?> userType = ClassUtils.getUserClass(handlerType);
    Map<Method, T> methods = MethodIntrospector.selectMethods(userType,
        (MethodIntrospector.MetadataLookup<T>) method -> {
          try {
            return getMappingForMethod(method, userType);
          // 略                                          }
```

https://github.com/spring-projects/spring-framework/blob/a0c97e4c36e5e07bc13bab4409ec740332a57871/spring-webmvc/src/main/java/org/springframework/web/servlet/handler/AbstractHandlerMethodMapping.java#L275

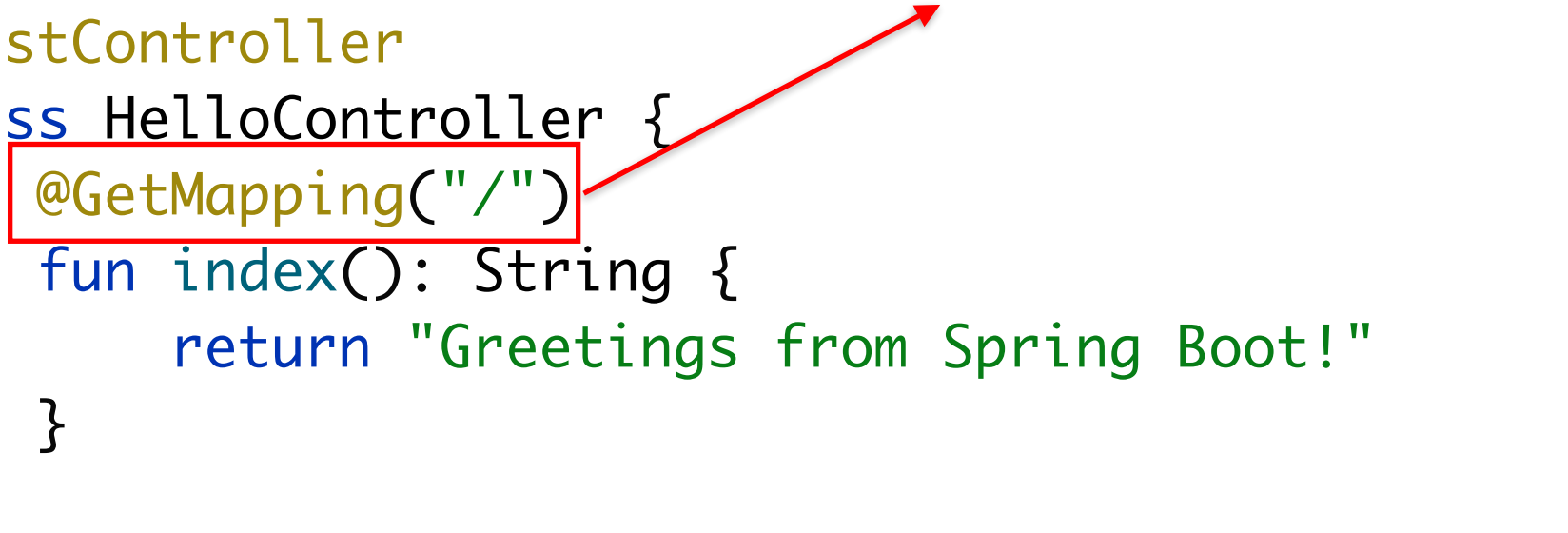
MethodIntrospector.selectMethods



https://github.com/spring-projects/spring-framework/blob/main/spring-core/src/main/java/org/springframework/core/MethodIntrospector.java#L84

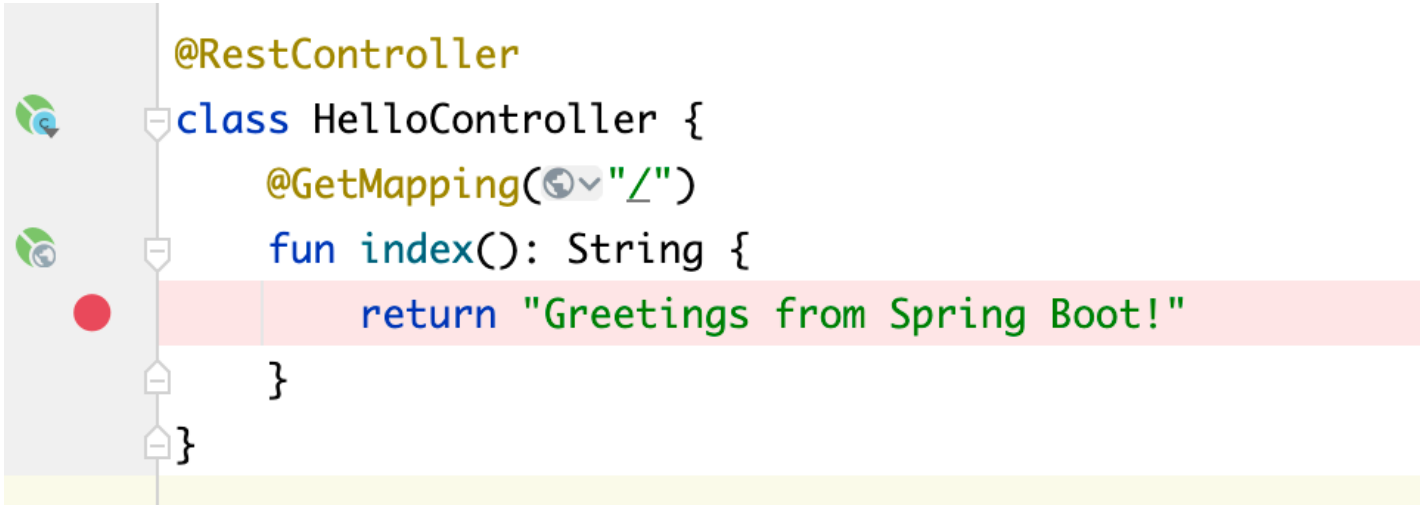
AutoConfigurationが起動時にマッピングする設定を返している

```
HelloController$index -> GET [/]
```



```
@RestController
class HelloController {
    @GetMapping("/")
    fun index(): String {
        return "Greetings from Spring Boot!"
    }
}
```

あとは実行時がどうなっているか見る



InvocableHandlerMethodで関数を実行している

```
@Nullable
protected Object doInvoke(Object... args) throws Exception {   args: Object[0]@7307
    Method method = getBridgedMethod();   method: "public java.lang.String com.exampl
    try {
        if (KotlinDetector.isSuspendingFunction(method)) {
            return CoroutinesUtils.invokeSuspendingFunction(method, getBean(), args)
        }
        return method.invoke(getBean(), args);   args: Object[0]@7307    method: "pub
    }
    catch (IllegalArgumentException ex) {
        assertTargetBean(method, getBean(), args);
        String text = (ex.getMessage() != null ? ex.getMessage() : "Illegal argument'
        throw new IllegalStateException(formatInvokeError(text, args), ex);
    }
```

- HttpServletRequestとRequestMappingHandlerAdapterを使って関数を取り出して実行する
- レスポンスを処理する箇所などにもたどり着ける

https://github.com/spring-projects/spring-framework/blob/main/spring-web/src/main/java/org/springframework/web/method/support/InvocableHandlerMethod.java#L205

寄り道

```
@Nullable
protected Object doInvoke(Object... args) throws Exception {   args: Object[0]@7307
    Method method = getBridgedMethod();   method: "public java.lang.String com.exampl
    try {
        if (KotlinDetector.isSuspendingFunction(method)) {
            return CoroutinesUtils.invokeSuspendingFunction(method, getBean(), args)
        }
        return method.invoke(getBean(), args);   args: Object[0]@7307    method: "pub
    }
    catch (IllegalArgumentException ex) {
        assertTargetBean(method, getBean(), args);
        String text = (ex.getMessage() != null ? ex.getMessage() : "Illegal argument'
        throw new IllegalStateException(formatInvokeError(text, args), ex);
    }
```

suspendにしてみる

```
@RestController
class HelloController {
    @GetMapping("/")
    suspend fun index(): String {
        return "Greetings from Spring Boot! $coroutineContext"
    }
}
```

めっちゃエラーでコケるが...

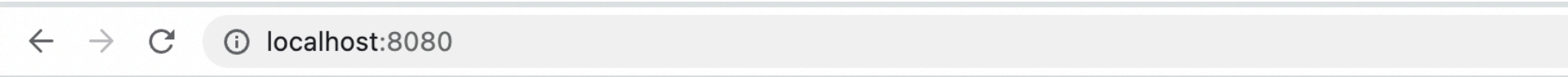
```
2022-02-08 18:56:13.857 ERROR 3052 --- [nio-8080-exec-4] o.a.c.c.C.[.[.[/].[dispatcherServlet]

java.lang.ClassNotFoundException Create breakpoint : kotlinx.coroutines.Dispatchers
    at org.springframework.core.CoroutinesUtils.invokeSuspendingFunction(CoroutinesUtils.java:74)
    at javax.servlet.http.HttpServlet.service(HttpServlet.java:655) ~[tomcat-embed-core-9.0.56.jar
    at javax.servlet.http.HttpServlet.service(HttpServlet.java:764) ~[tomcat-embed-core-9.0.56.jar
```

めっちゃエラーでコケるが...

```
implementation(“io.projectreactor.kotlin:reactor-kotlin-extensions")
implementation("org.jetbrains.kotlinx:kotlinx-coroutines-reactor")
```

必要なライブラリを追加すればいけるのでは？

いける



Greetings from Spring Boot! [Context0{}, MonoCoroutine{Active}@28edf533, Dispatchers.Unconfined]

まとめ

- Spring Framework/BootのソースはGithubにある。IntelliJで開ける
- Spring Frameworkがコンポーネントスキャンでインスタンスを作ったり初期化したりDIしていて、起動プロセスを読んでいくとわかる
- Spring Bootがモジュールの初期設定系をAuto Configurationによって行っている。勝手に動き出す系は大体Auto Configurationを探せば見つかる
- @RestControllerが付いたクラスはコンポーネントスキャンでインスタンスが作られ、RequestMappingHandlerMappingによって関数とパスとメソッドが紐付けられる
- リクエストを受けて関数を呼び出しているのはInvocableHandlerMethod。ブレークポイント張れば動かしながら追える

Thank You