Ruby 2.0 on Rails

RUBYCONF 2012 DENVER COLORADO

@a_matsuda

Ruby 2.0 (on Rails)

whoami

- Akira Matsuda
  - GitHub: amatsuda
  - Twitter: @a_matsuda
- CRuby core since 1.year.ago

Asakusa.rb

http://www.flickr.com/photos/takkanm/3978417669

begin

Ruby 2.0

Ruby 2.0

- 🎃 "100% compatible"

Ruby 2.0

- "100% compatible" (?)

Ruby 2.0

- "100% compatible"
- 100% awesome new features!

Ruby 2.0

🎃 "100% compatible"

+

🎃 100% awesome new features!

=

🎃🎃 Ruby 2.0.0!

New Features

New Features

- "feature freeze" - 1.week.ago

New Features

- Refinements
- Module#prepend
- Enumerable#lazy
- Keyword Arguments

Refinements

@shugomaeda

@shugomaeda

- Matz's boss at NaCl
  - http://www.netlab.jp/
- Ruby Association
  - http://www.ruby.or.jp

Asakusa.rb

http://www.flickr.com/photos/takkanm/3978417669

What are Refinements?

- A mechanism to extend Classes and Modules locally
  - Lexically scoped
  - Safer and cleaner than open-class monkey patching

background

- Monkey patching affects all instances

Monkey patching affects all instances

```ruby
class Person; attr_accessor :name; end
matz = Person.new
matz.name = 'matz'
matz.age
#=> NoMethodError: undefined method `age'

class Person; attr_accessor :age; end
shugo = Person.new
shugo.name = 'shugo'
shugo.age = 34

matz.age
#=> nil
```

Refinements

Refinements

- Module#refine
- Kernel#using

Module#refine

Module#refine

```ruby
module Foo
  refine String do
    def say
      puts "#{self}!!"
    end
  end

  'hello'.say
end
#=> hello!!
```

Module#refine

```
module Foo
  refine String do
    def say; puts "#{self}!!"; end
  end

  'hello'.say
end
#=> hello!!

'world'.say
#=> NoMethodError: undefined method `say'
for "world":String
```

Kernel#using

using Kernel#using

```ruby
module Foo
  refine String do
    def say; puts "#{self}!!"; end
  end
end

class Bar
  using Foo

  def hello
    'hello'.say
  end
end

Bar.new.hello
#=> hello!!
```

using using in Module

```
module Foo
  refine String do
    def say; puts "#{self}!!"; end
  end
end

module Bar
  using Foo
end

Bar.module_eval {
  'hello'.say
}
#=> hello!!
```

using anonymous Module

```ruby
module Foo
  refine String do
    def say; puts "#{self}!!"; end
  end
end

Module.new { using Foo }.module_eval {
  'hello'.say
}
#=> hello!!

'world'.say
#=> NoMethodError: undefined method `say' for
"world":String
```

using lambda / Proc?

```
module Foo
  refine String do
    def say; puts "#{self}!!"; end
end; end

-> {
  using Foo
  'hello'.say
}.call
#=> hello!!

'world'.say
#=> world!!
```

using lambda + Module

```
module Foo
  refine String do
    def say; puts "#{self}!!"; end
end; end

Module.new { using Foo }.module_eval {
  -> { 'hello'.say }
}.call
#=> hello!!

'world'.say
#=> undefined method `say' for
"world":String
```

examples

Refinements example (1)

- rspec-refinements

background

RSpec

```
describe RSpec do
  its(:syntax) {
    should be_elegant
  }
end
```

rspec-expectations/lib/rspec/expectations/syntax.rb (edited)

```
module RSpec::Expectations::Syntax
  def enable_should(syntax_host = default_should_host)
    BasicObject.module_eval do
      def should(matcher=nil, message=nil, &block)
        ::RSpec::Expectations::PositiveExpectationHand
ler.handle_matcher(self, matcher, message, &block)
      end

      def should_not(matcher=nil, message=nil, &block)
        ::RSpec::Expectations::NegativeExpectationHand
ler.handle_matcher(self, matcher, message, &block)
      end
    end
  end
end
```

global pollution

```
Object.new.should
#=> NoMethodError: undefined method
`should'

require 'rspec'
Object.new.should
=>
#<RSpec::Matchers::BuiltIn::Positiv
eOperatorMatcher:0x007fd8910c0528
@actual=#<Object:0x007fd8910c0550>>
```

rspec-refinements

- https://github.com/amatsuda/rspec-refinements
- % gem i rspec-refinements

rspec-refinements.rb

```
require 'rspec'

BasicObject.class_eval do
  undef :should
  undef :should_not
end
                                                # copied from
module RSpec::Refinements::ExampleMethods       rspec-expectation/
  refine BasicObject do                         expectations/syntax.rb
    def should(matcher=nil, message=nil, &block)
      ::RSpec::Expectations::PositiveExpectationHandler.handle_matcher(self,
matcher, message, &block)
    end

    def should_not(matcher=nil, message=nil, &block)
      ::RSpec::Expectations::NegativeExpectationHandler.handle_matcher(self,
matcher, message, &block)
ennnd

RSpec.configure do |c|
  c.before :all do                             # ugly but works...
    RSpec.world.example_groups.each do |eg|
      eg.send :using, RSpec::Refinements::ExampleMethods
ennnd
```

rspec-refinements/spec/foo_spec.rb

```ruby
class Foo
  def tes
    self.should
  end
end

describe Foo do
  # make sure the `should` method actually works inside an
Example
  it { should be_a Foo }

  specify {
    expect { Foo.new.tes }.to raise_error(NoMethodError, /
\Aundefined method `should' for #<Foo:/)
  }
end
```

Refinements example (2)

- activerecord-refinements

Feature

```
User.where('age >= ?', 18)
```

Feature

```
User.where('age >= ?', 18)

=>

User.where { :age >= 18 }
```

activerecord-refinements

- https://github.com/amatsuda/activerecord-refinements
- % gem i activerecord-refinements

activerecord-refinements/lib/active_record/refinements.rb (1)

```
module ActiveRecord::Refinements
  module WhereBlockSyntax
    refine Symbol do
      %i[== != =~ > >= < <=].each
do |op|
        define_method(op) {|val|
[self, op, val] }
ennnnd
```

activerecord-refinements/lib/active_record/refinements.rb (2)

```ruby
module ActiveRecord::Refinements
  module QueryMethods
    def where(opts = nil, *rest, &block)
      if block
        col, op, val = Module.new { using
ActiveRecord::Refinements::WhereBlockSyntax }.module_eval &block
        arel_node = case op
        when :!=
          table[col].not_eq val
        when :=~
          table[col].matches val
        when :>
          table[col].gt val
        when ...  # (snip)
        end

        clone.tap do |relation|
          relation.where_values += build_where(arel_node)
        end
      else
        super
ennnnd
```

activerecord-refinements/lib/activerecord-refinements.rb

```
module ActiveRecord::QueryMethods
  prepend ActiveRecord::Refinements::QueryMethods
end
```

activerecord-refinements/ spec/where_spec.rb

```ruby
describe 'Symbol enhancements' do
  describe '#!=' do
    subject { User.where { :name != 'nobu' }.to_sql }
    it { should =~ /WHERE \("users"."name" != 'nobu'\)/ }
  end

  describe '#>=' do
    subject { User.where { :age >= 18 }.to_sql }
    it { should =~ /WHERE \("users"."age" >= 18\)/ }
  end

  describe '#=~' do
    subject { User.where { :name =~ 'tender%' }.to_sql }
    it { should =~ /WHERE \("users"."name" LIKE 'tender%'\)/ }
  end

  context 'outside of where block' do
    specify {
      expect { :omg > 1 }.to raise_error ArgumentError
    }
  end
end
```

Refinements example (3)

- activesupport-refinements

background

ActiveSupport

- core_ext
- others

core_ext

```
class Numeric
  def days
    ActiveSupport::Duration.new(self * 24.hours, [[:days, self]])
  end
end

p 3.days
#=> 259200


class String
  def blank?
    self !~ /[^[:space:]]/
  end
end

p '  '.blank?
#=> true
```

"No more free lunch"

- https://github.com/rails/rails/commit/ab32126
- "require 'active_support' no longer orders the whole menu of core extensions. Ask for just what you need: e.g. require 'active_support/core/time' to use timezones, durations, and stdlib date/time extensions. [Jeremy Kemper]"

The AS 3 way

```
require 'active_support'
require 'active_support/core_ext/
object/try'

p 1234.try :to_s
#=> "1234"
p 1234.days
#=> undefined method `days' for
1234:Fixnum (NoMethodError)
```

A dangerous controller

```
require 'active_support/core_ext/numeric/time'

class FooController < ApplicationController
  def index
    @foos = Foo.where { :created_at >
3.days.ago }
  end
end
```

activesupport-refinements

- https://github.com/amatsuda/activesupport-refinements
- % gem i activesupport-refinements

activesupport-refinements/spec/try_spec.rb

```ruby
require 'active_support/refinements/core_ext/object/try'

describe 'Object#try' do
  context 'when using ObjectExt::Try' do
    it 'an Object has #try method' do
      Module.new { using ObjectExt::Try }.module_eval
{ 'hello'.try(:reverse) }.should == 'olleh'
    end
  end

  context 'when not using ObjectExt::Try' do
    it 'has no #try method' do
      expect { 'hello'.try(:reverse) }.to raise_error
NoMethodError
    end
  end
end
```

A safer controller

```
require 'active_support/refinements/
core_ext/numeric/time'

class FooController < ApplicationController
  using NumericExt::Time

  def index
    @foos = Foo.where { :created_at >
3.days.ago }
  end
end
```

WARNING

Refinements are experimental, and may be removed from Ruby 2.0!

Module#prepend

background

Ruby < 2, Rails < 3

alias_method_chain

```
def alias_method_chain(target, feature)
  alias_method "#{target}_without_#{feature}", target
  alias_method target, "#{target}_with_#{feature}"
end
```

https://github.com/rails/rails/commit/794d93f by @jamis

AMC

```
class A
  def foo; puts 'foo'; end
end

class A
  def foo_with_bar
    foo_without_bar
    puts 'bar'
  end
  alias_method_chain :foo, :bar
end

A.new.foo

A.new.foo_without_bar
```

simple pagination

```
ActiveRecord::FinderMethods.module_eval do
  def all_with_page(*args)
    if args.any? && (page =
args.first.delete(:page))
      limit(10).
        offset(10 * (page - 1)).
        all_without_page(*args)
    else
      all_without_page(*args)
    end
  end

  alias_method_chain :all, :page
end
```

Problem

adding `baz` and calling foo "without_bar"

```
class A
  def foo; puts 'foo'; end

  def foo_with_bar
    foo_without_bar
    puts 'bar'
  end
  alias_method_chain :foo, :bar

  def foo_with_baz
    foo_without_baz
    puts 'baz'
  end
  alias_method_chain :foo, :baz
end

A.new.foo_without_bar
```

"without_bar" skips "baz"

```
class A
  def foo; puts 'foo'; end

  def foo_with_bar
    foo_without_bar
    puts 'bar'
  end
  alias_method_chain :foo, :bar

  def foo_with_baz
    foo_without_baz
    puts 'baz'
  end
  alias_method_chain :foo, :baz
end

A.new.foo_without_bar
#=> foo
```

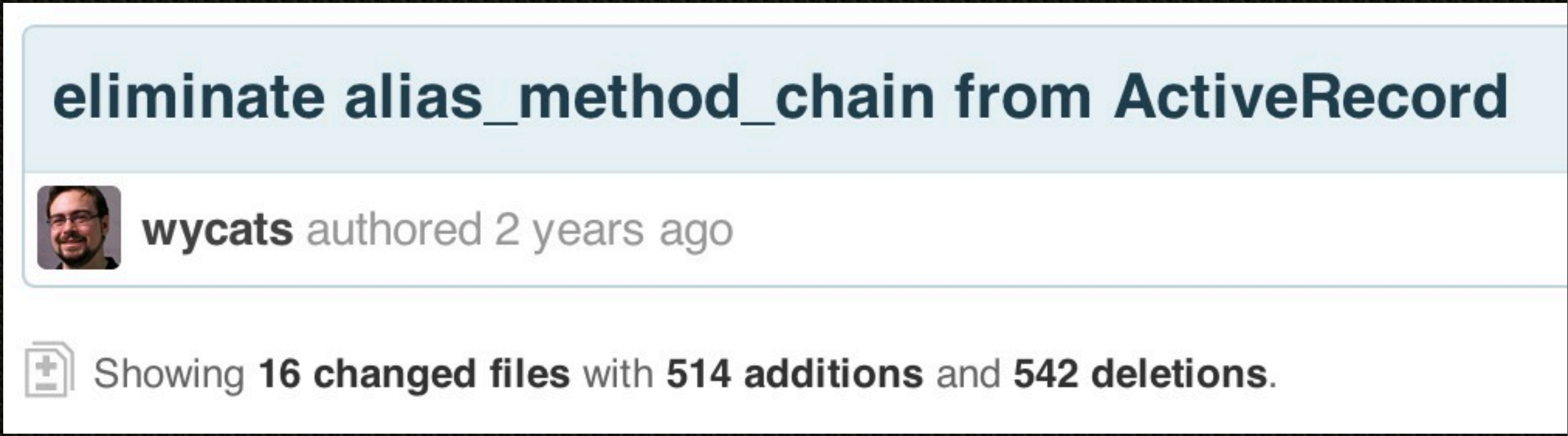
an actual example

```ruby
gem 'activerecord', '<2.3'
require 'active_record'

ActiveRecord::Base.configurations = {'test' => {:adapter =>
'sqlite3', :database => ':memory:'}}
ActiveRecord::Base.establish_connection('test')

class User < ActiveRecord::Base
  validates_presence_of :name
end

class CreateAllTables < ActiveRecord::Migration
  def self.up
    create_table(:users) {|t| t.column :name, :string}
  end
end
CreateAllTables.up

# p User.new.save_without_validation
p User.new.save_without_dirty
p User.new.save
```

Rails 3’s solution

- use the power of Ruby Module
- super

https://github.com/rails/rails/commit/d916c62



eliminate alias_method_chain from ActiveRecord

wycats authored 2 years ago

Showing **16 changed files** with **514 additions** and **542 deletions**.

excerpt from d916c62

```
included do
- alias_method_chain :save, :dirty
  ...
end

-   def save_with_dirty(*args)
-     if status =
save_without_dirty(*args)
+   def save(*) #:nodoc:
+     if status = super
  ...
```

@wycats style

```ruby
module Bar
  def foo
    puts 'bar'
    super
end; end

module Baz
  def foo
    puts 'baz'
end; end

class A
  include Baz
  include Bar

  def foo
    puts 'foo'
    super
  end
end

A.new.foo
#=> foo
bar
baz
```

the original foo method

```
class A
  def foo
    puts 'foo'
    super
  end
end
```

how can we extend methods that are not calling super?

```
class A
  def foo
    puts 'foo'
  end
end

(do something...)


A.new.foo
#=> foo
bar
```

AMC?

```
class A
  def foo
    puts 'foo'
  end
end

class A
  def foo_with_bar
    foo_without_bar
    puts ‘bar’
  end
  alias_method_chain :foo, :bar
end

A.new.foo
#=> foo
bar
```

Module#prepend

@wycats
@n0kada

Nobu

The Patch Monster

```
% git clone git://github.com/ruby/ruby

% git shortlog -ns | grep -v svn | head -10
  7049 nobu
  3470 akr
  2553 matz
  1492 naruse
  1464 usa
  1412 eban
   746 kol
   580 knu
   531 mame
   513 drbrain
```

Who's creating Ruby?

- Matz: designing
- Ko1: implementing
- Nobu: fixing

Who's creating Ruby?

- 🎃 Matz: designing
- 🎃 Ko1: implementing
- 🎃 Nobu: fixing

heroku

Nobu

Asakusa.rb

http://www.flickr.com/photos/takkanm/3978417669

AMC

```
class A
  def foo
    puts 'foo'
  end
end

class A
  def foo_with_bar
    foo_without_bar
    puts 'bar'
  end
  alias_method_chain :foo, :bar
end

A.new.foo
#=> foo
bar
```

Module#prepend

```
class A
  def foo; puts 'foo'; end
end

module Bar
  def foo
    super
    puts 'bar'
  end
end

class A
  prepend Bar
end

A.new.foo
#=> foo
bar
```

Module#prepend

```
class A; def foo; puts 'foo'; end; end

module Bar
  def foo
    super
    puts 'bar'
  end
end

module Baz
  def foo
    super
    puts 'baz'
  end
end

class A
  prepend Bar
  prepend Baz
end

A.new.foo
#=> foo
bar
baz
```

simple pagination with Module#prepend

```
module PrependPaginator
  def all(*args)
    if args.any? && (page =
args.first.delete(:page))
      self.limit_value = 10
      self.offset_value = 10 * (page - 1)
    end
    super
  end
end

ActiveRecord::FinderMethods.send :prepend,
PrependPaginator
```

activerecord-refinements/lib/activerecord-refinements.rb

```
module ActiveRecord::Refinements
  module QueryMethods
    def where(opts = nil, *rest, &block)
      if block
        ...   # (snip)
      else
        super
ennnnd

module ActiveRecord::QueryMethods
  prepend ActiveRecord::Refinements::QueryMethods
end
```

Enumerable#lazy

@yhara

まつもとゆきひろ
コードの世界
スーパープログラマになる14の思考法
言語開発者がプログラミングの本質を説く

Redmine #4890

Enumerator

- [1, 2, 3].each
- => #<Enumerator: [1, 2, 3]:each>

Enumerator::Lazy

- [1, 2, 3].lazy
- => #<Enumerator::Lazy: [1, 2, 3]>

lazy evaluation

```
(1..Float::INFINITY).lazy
  .select(&:even?)
  .take(20)
  .force
#=> [2, 4, 6, 8, 10, 12, 14, 16,
18, 20, 22, 24, 26, 28, 30, 32,
34, 36, 38, 40]
```

Friday the 13th (until ETERNITY)

```
require 'date'

ETERNITY = Float::INFINITY

puts (Date.today..ETERNITY).
  lazy.
  select {|d| (d.day == 13) && d.friday?}.
  take(5).force
#=> 2013-09-13
2013-12-13
2014-06-13
2015-02-13
2015-03-13
```

Ruby implementation of UNIX wc command

```
(ARGV.length == 0 ?
 [["", STDIN]] :
 ARGV.lazy.map { |filename|
   [filename, File.open(filename)]
}).map { |filename, file|
   "%4d  %4d  %4d %s\n" % [*file.lines.lazy.map { |line|
     [1, line.split.length, line.length]
   }.inject([0, 0, 0]) { |(lc, wc, cc), (l, w, c)|
     [wc + w, lc + l, cc + c]
   }, filename]
}.each(&:display)
```

by @shugomaeda
http://shugomaeda.blogspot.com/2012/03/enumerablelazy-and-its-benefits.html

Keyword Arguments

@mametter

@mametter

- Quine artist
- Ruby 2.0 release manager

background

Well known idiom using Hash

```ruby
def foo(args = {})
  puts "a is #{args[:a]}, b is #{args[:b]}"
end

foo a: 1, b: 2
#=> a is 1, b is 2
```

Patterns & Implementations

- Default Values
- Multiple Hashes
- Splat Operator
- Assert Valid Keys

Default Values

merge

```
def foo(args = {})
  values = {a: 1, b: 2}.merge args
  puts "a is #{values[:a]}, b is #{values[:b]}"
end

foo b: 4
#=> a is 1, b is 4
```

reverse_merge (ActiveSupport)

```ruby
def foo(args = {})
  values = args.reverse_merge a: 1, b: 2
  puts "a is #{values[:a]}, b is #{values[:b]}"
end

foo b: 4
#=> a is 1, b is 4
```

multiple Hashes

- **def create(attributes = {}, options = {}, &block)**
  AR/associations/collection_association.rb
- **def form_tag(url_for_options = {}, options = {}, &block)**
  AV/helpers/form_tag_helper.rb
- **def render(options = {}, locals = {}, &block)**
  AV/helpers/rendering_helper.rb
- **def button_to(name, options = {}, html_options = {})**
  AV/helpers/url_helper.rb
- **def date_select(method, options = {}, html_options = {})**
  AV/helpers/date_helper.rb

Passing values into the latter keywords Hash

```
def button_to(name, options = {}, html_options = {})

<%= button_to 'New!', action: 'new', method: 'get' %>
#=> options: {action: 'new', method: 'get'},
    html_options: {}

<%= button_to 'New!', {action: 'new'}, {method: 'get'} %>
#=> options: {action: 'new'},
    html_options: {method: 'get'}
```

Splat Operator

```
def define_model_callbacks(*callbacks)
  options = callbacks.extract_options!
  options = {
    :terminator => "result == false",
    :scope => [:kind, :name],
    :only => [:before, :around, :after]
  }.merge(options)
  ...
```

AMo/callbacks.rb

extract_options!

```
class Array
  def extract_options!
    if last.is_a?(Hash) &&
last.extractable_options?
      pop
    else
      {}
    end
  end
end
```

AS/core_ext/array/extract_options.rb

assert_valid_keys

```
VALID_FIND_OPTIONS =
[:conditions, :include, :joins, :limit, :offset,
  :extend, :order, :select, :readonly,
  :group, :having, :from, :lock ]

def apply_finder_options(options)
  ...
  options.assert_valid_keys(VALID_FIND_OPTIONS)
```

AR/relation/spawn_methods.rb

Implementation of assert_valid_keys

```
def assert_valid_keys(*valid_keys)
  valid_keys.flatten!
  each_key do |k|
    raise ArgumentError.new("Unknown key: 
#{k}") unless valid_keys.include?(k)
  end
end
```

AS/core_ext/hash/keys.rb

using assert_valid_keys

```
{name: 'amatsuda', job: 'Rubyist'}
  .assert_valid_keys(:name, :age)
#=> ArgumentError: Unknown key: job
```

Ruby 2.0 keyword arguments

Defalut Values

No need to merge or reverse_merge anymore!

```
def foo(a: 1, b: 2)
  puts "a is #{a}, b is #{b}"
end


foo()
#=> a is 1, b is 2


foo a: 3, b: 2
#=> a is 3, b is 2
```

Multiple Hashes & Splat Operator

The "rest" argument

```
def button_to(name, controller: nil, action: nil,
**html_options)
  puts "controller: #{controller}"
  puts "action: #{action}"
  puts "html_options: #{html_options}"
end

button_to 'a', action: 'new', method: 'get'
#=> controller: (nil)
action: new
html_options: {:method=>"get"}
```

assert_valid_keys

We don't need assert_valid_keys anymore!

```
def apply_finder_options(conditions: nil,
  include: nil, joins: nil, limit: nil, offset: nil,
  extend: nil, order: nil, select: nil,
  readonly: nil, group: nil, having: nil,
  from: nil, lock: nil)

  puts "order: #{order}"
  puts "limit: #{limit}"
end

apply_finder_options order: 'id', limit: 3, omg: 999
#=> unknown keyword: omg (ArgumentError)
```

ensure

try 2.0

```
if your_app.ruby_version == 1.9.3

  you.should upgrade {|ruby|

    your_app << ruby.new_features!

    raise Issue.new unless ruby.compatible?

  }

end
```

try 2.0

```
if your_app.ruby_version == 1.9.3

  you.should upgrade {|ruby|

    your_app << ruby.new_features!

    raise Issue.new unless ruby.compatible?

  }

else
  die
end
```

Use the new features

- 🎃 Because it's useful
- 🎃 In order to speed up transition
- 🎃 And let's deprecate 1.9.3 ASAP!

end