メソッドを操る

Control methods like a pro

A guide to Ruby's awesomeness, a.k.a. metaprogramming

OKURA Masafumi, RubyWorld Conference 2021

Methods

Do you want to...

- Add conventions to the order of method invocations
  - e.g. MiniTest to call methods starting with “test” automatically
- Modify existing methods without overhead
  - e.g. Something like ActiveSupport::Callbacks but without any performance penalty

Example1: Superclass for abstract logic

```ruby
class Logic
  def call
    (methods + private_methods).select do |method_name|
      method_name.start_with?('validate_')
    end.each do |method_name|
      m = method(method_name)
      puts "#{m.name} is executed"
      m.call
    end
    main
  end
end
```

Example2: Subclass for concrete logic

```ruby
require_relative 'logic'
class MyLogic < Logic
  def initialize(name = '')
    @name = name
  end

  def main
    puts @name
  end

  private

  def validate_name_presence
    raise 'name is not present' if @name.nil? || @name.empty?
  end

  def validate_name_length
    raise 'name is too long' if @name.length >= 10
    raise 'name is too short' if @name.length <= 2
  end
end
```

Example 3: When we execute a concrete logic

```
❯ irb
irb(main):3.0.2:001:0> require_relative 'my_logic'
true
irb(main):3.0.2:002:0> MyLogic.new('Masafumi').call
validate_name_presence is executed
validate_name_length is executed
Masafumi
nil
irb(main):3.0.2:003:0> MyLogic.new.call
validate_name_presence is executed
/Users/okuramasafumi/Sandbox/Ruby/control_methods_like_a_pro/my_logic.rb:14:in `validate_name_presen
ce': name is not present (RuntimeError)
```

pp @okuramasafumi

- Name: **OKURA Masafumi** (Masafumi is my first name :D)
- Ruby experience: **since 2012**
- Work as: Freelance Ruby/Rails dev, tutor
- Organizer of: **Kaigi on Rails** (https://kaigionrails.org)
- Creator of: **Alba** gem (JSON serializer, https://github.com/okuramasafumi/alba) along with a few others

Part1: Know

Methods for methods

Methods that list methods

- Note: these methods return the method name as a Symbol, not the method object
  - `**methods**` for listing public and protected methods
  - `**private_methods**` for listing private methods
  - `**singleton_methods**` for listing singleton methods, practically used to list class methods

Methods that fetch method

- `**method**` for fetching a method object with a given name from an object
  - e.g. `**'foo'.method(:gsub)**` returns callable/executable Method
- `**instance_method**` for fetching a method object with a given name from a class
  - e.g. `**String.instance_method(:gsub)**` does similar, but the returned object is UnboundMethod that's not callable

Method object

Method class

- Associated with a particular object, not only a class
- **Callable**
- Can be converted into a Proc with `to_proc`
- Can be converted into an UnboundMethod with `unbind`

UnboundMethod class

- Not associated with an object
- **Not callable**
- Cannot be converted into a Proc since Proc should be callable
- Can be converted into Method with `bind`

Inspect

- `name`
- `parameters`
- `arity`
- `source_location`
- ~~`body`~~

Demo

Part2: Define

Define methods

- Using `**def**` keyword
  - Simple
  - Static
- Using `**define_method**` method
  - Dynamic
  - Can be used with Proc and Method object as a method body

Undefine methods

- Both `**undef**` keyword and `**undef_method**` are quite similar
  - They both prohibit an object to respond
  - `**undef_method**` is more dynamic
- `**remove_method**` just removes a method from an object
  - When a parent class responds to that method, that will be called

Redefine methods

1. Decide the name of the target
2. Fetch method object using `**method**`
3. Create a new Proc inside which fetched method object is called before/after some extra bit
4. Remove a method using `**remove_method**`
5. Define a new method with the same name using `**define_method**` with a newly created Proc as a method body

Demo

Conclusion

- In Ruby, methods are objects
- You can play with them, it’s not scary!
- Metaprogramming gives us the power to do awesome things
- Join us!

Next step

- https://github.com/okuramasafumi/tiny_hooks
  - The repository of the second demo, has some nice tricks
- https://docs.ruby-lang.org/en/
  - Official document
- And your code!