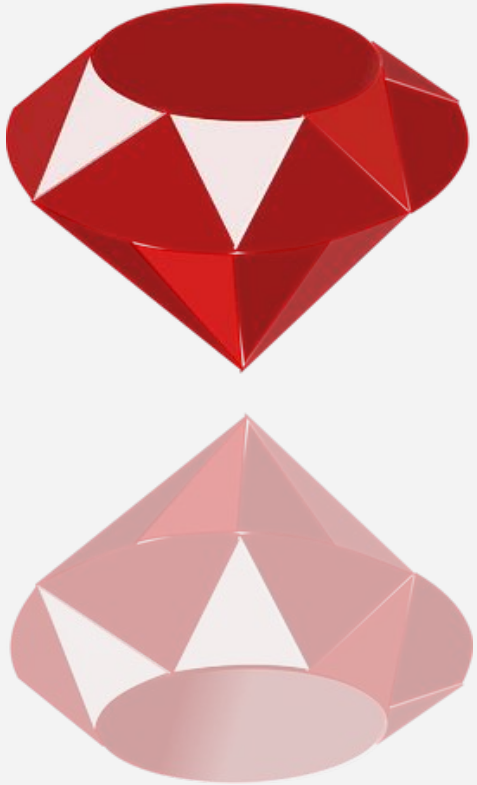
Yalantis

Лекция 7: Метапрограммирование

Курс лекций по основами web-разработки на языке программирования Ruby

Отражение

(рефлексия, интроспекция, самоанализ) - способность программы исследовать свое состояние и свою структуру



Кто я ?

```
num = 1.0
num.object_id # -36028797018963966
num.class # Integer
num.class.class # Classs
num.class.superclass # Numeric
num.is_a? Float # true
num.is_a? Numeric # true
num.kind_of? Float # true
num.kind_of? Numeric # true
num.instance_of? Float # true
num.instance_of? Numeric # false
num.respond_to?(:to_s) # true
```

```
module A; end
module B; include A; end;
class C
 include B
end

C < B
# true
B < A
# true
C < A
# true
Float < Integer
# nil
Integer < Comparable
# true
Integer < Float
# nil
String < Numeric
# nil
```

```
String.ancestors
#  [String, Comparable, Object, Kernel,
BasicObject]
A.ancestors
# [A]
B.ancestors
# [B, A]
C.ancestors
# [C, B, Object, A, Kernel,
BasicObject]
```

Исследуем класс

```ruby
class Octopus
 @@octo_var = 2
 TENTACLES = 8

 def initialize(n)
  @name = n
 end

 def speak
  puts "I'm an octopus named #{@name}"
 end

 def Octopus.classgreeting
  puts "Hi from class Octopus"
 end

 private

 def private_method
  "Hello Yalantis !"
 end
end
```

```ruby
Octopus.private_instance_methods(false)
# [:initialize, :private_method]
Octopus.private_instance_methods
# [:initialize, :DelegateClass, :sprintf ....
Octopus.public_instance_methods(false)
Octopus.public_instance_methods(true)
# включая предков
Octopus.constants
# [:TENTACLES]
Octopus.class_variables
# [:@@octo_var]
Octopus.singleton_methods
# [:classgreeting]
```

ObjectSpace

```ruby
a = Regexp.new
b = Regexp.new
count = ObjectSpace.each_object(Regexp) {|x| p x }
puts "Total count: #{count}"
# /irb\/.*\.rb/
# /\/tmp\/irb-binding/
# /\(irb_local_binding\)/
# /%([0-9]+)?([a-zA-Z])/
# /\A_?/
# ....
# Total count: 269

count = ObjectSpace.each_object(Array) {|x| p x }
# Total count: 1798

a = 1
b = 2
count = ObjectSpace.each_object(Integer) {|x| p x }
puts "Total count: #{count}"

#~ ruby rubyroid.rb
# 9223372036854775807
# 337822478979798833673471578767478577 87
# Total count: 2
```

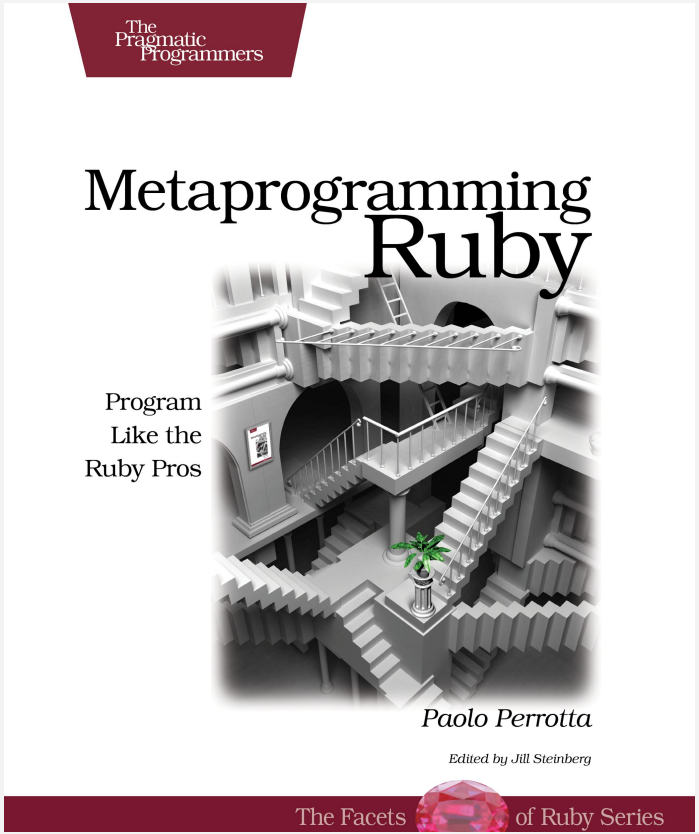
```ruby
h = {}
ObjectSpace.count_objects(h)
puts h
{:TOTAL=>44024, :FREE=>7052, :T_OBJECT=>529,
:T_CLASS=>1164, :T_MODULE=>62, :T_FLOAT=>4,
:T_STRING=>22905, :T_REGEXP=>268, :T_ARRAY=>1976,
:T_HASH=>1880, :T_STRUCT=>17, :T_BIGNUM=>4,
:T_FILE=>5, :T_DATA=>266, :T_MATCH=>60, :T_COMPLEX=>1,
:T_SYMBOL=>30, :T_IMEMO=>7725, :T_ICLASS=>76}
```

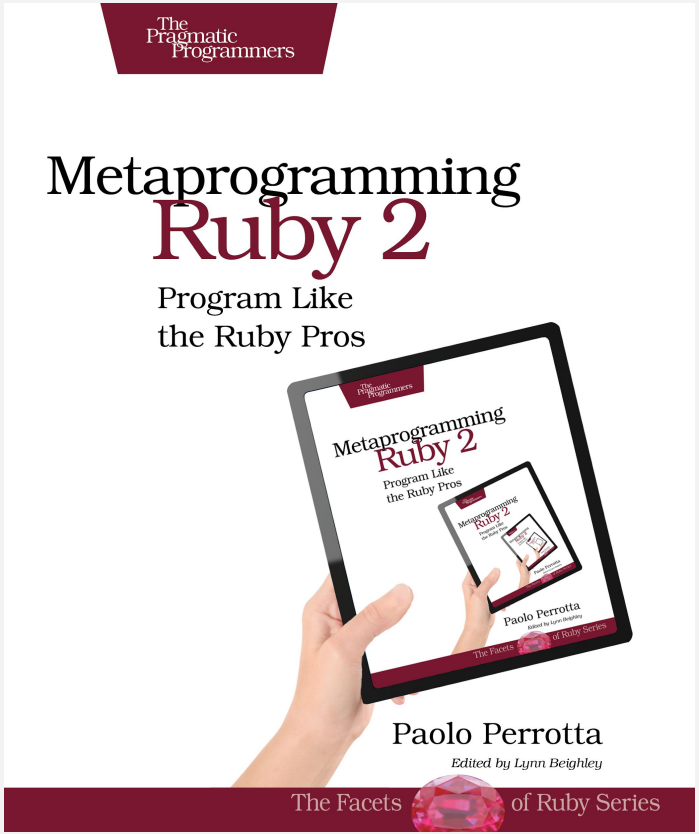
Трассировка

```ruby
trace = TracePoint.new(:raise) do |tp|
 p [tp.lineno, tp.event, tp.raised_exception]
end
#=> #<TracePoint:disabled>

trace.enable
#=> false

0 / 0
#=> [9, :raise, #<ZeroDivisionError: divided by 0>]
```

```ruby
def func1
 puts caller(0)
end

def func2
  func1
end

func2
# ...lib/ruby/2.6.0/irb/workspace.rb:85:in `eval'
# ...lib/ruby/2.6.0/irb/workspace.rb:85:in `evaluate'
# ...lib/ruby/2.6.0/irb/context.rb:385:in `evaluate'
# ...lib/ruby/2.6.0/irb.rb:493:in `block (2 levels) in eval_input'
# ...lib/ruby/2.6.0/irb.rb:647:in `signal_status'
# ...lib/ruby/2.6.0/irb.rb:490:in `block in eval_input'
# ...lib/ruby/2.6.0/irb/ruby-lex.rb:246:in `block (2 levels) in each_top_level_statement'
# ...lib/ruby/2.6.0/irb/ruby-lex.rb:232:in `loop'
# ...lib/ruby/2.6.0/irb/ruby-lex.rb:232:in `block in each_top_level_statement'
# ...lib/ruby/2.6.0/irb/ruby-lex.rb:231:in `catch'
# ...lib/ruby/2.6.0/irb/ruby-lex.rb:231:in `each_top_level_statement'
# ...lib/ruby/2.6.0/irb.rb:489:in `eval_input'
# ...lib/ruby/2.6.0/irb.rb:428:in `block in run'
# ...lib/ruby/2.6.0/irb.rb:427:in `catch'
# ...lib/ruby/2.6.0/irb.rb:427:in `run'
# ...lib/ruby/2.6.0/irb.rb:383:in `start'
# ...lib/ruby/gems/2.6.0/gems/irb-1.0.0/exe/irb:11:in `<top (required)>'
# ...bin/irb:23:in `load'
# ...bin/irb:23:in `<main>'
```

Метапрограммирование





Определение классов

```
M = Module.new

C = Class.new

D = Class.new(C) do
 include M
end
```

Eval

```
eval("2+2")
# => 3

eval("def sum(*numbers, &block); block.call(numbers.inject(0) {|sum, i| sum + i}); end")
sum(1, 2, 2) { |x| x * 2 }
# => 10

def get_binding(str)
 return binding
end
str = "guys"
eval "'Hello ' + str"                              # => "Hello Guys"
eval "'Hello ' + str", get_binding("Yalantis")  # => "Hello Yalantis"
```

instance_eval, class_eval

```ruby
class Account
 attr_accessor :state

 def initialize(state)
  @state = state
 end
end

Account.instance_eval do
 def is_awesome?
  "it is truly awesome"
 end
end

Account.class_eval do
 def debit(amount)
  self.state = state - amount
 end
end

Account.is_awesome?
# => "it is truly awesome"
account = Account.new(20)
account.debit(3)
account.state
# => 17
```

```ruby
account.instance_eval do
 def credit(amount)
  self.state = state + amount
 end
end

account.credit(35)
account.state
# => 52

# Но если мы инициализируем новый экземпляр,
то получим ошибку
account = Account.new(6)
account.credit(15)
# NoMethodError (undefined method `credit' for
#<Account:0x0000000000d259b0 @state=6>)

def account.other_version_of_credit(amount)
 self.state = state + amount
end

account.other_version_of_credit(5)
# => 11

def Account.other_awesome
 "still awesome"
end

Account.other_awesome
# => "still awesome"
```

```ruby
Account.instance_eval("def third_awesome; 'more
awesome then awesome';end")
Account.third_awesome
# => "more awesome then awesome"

class User
 def initialize
  @group = "superuser"
 end
end

User.instance_eval("@group")
# nil

User.class_eval("@group")
# nil

User.new.instance_eval("@group")
# "superuser"

User.new.class_eval("@group")
# NoMethodError
```

instance_exec, class_exec

```ruby
class User
 attr_reader :group

 def initialize
   @group = "superuser"
 end
end

bob = User.new

bob.instance_eval("stranger") { |variable| @group = variable }
# => "ArgumentError (wrong number of arguments (given 1, expected 0))"

bob = User.new

bob.instance_exec("stranger") { |variable| @group = variable }
bob.group
# => "stranger"
```

Переменные и константы

```ruby
Math.constants
# [:DomainError, :PI, :E]
Math.const_get(:PI)
# => 3.141592653589793
Math.const_defined? :PI
# => true

MY_CONST = "me"

Object.const_get("MY_CONST")
# => "me"

class MyClassMyRules
 def hah; "hah" end
end

Object.const_get("MyClassMyRules").new.hah
# => "hah"
```

```ruby
o = Object.new
o.instance_variable_set(:@x, 0)
o.instance_variable_get(:@x)
# => 0
o.instance_variable_defined?(:@x)
# => true

Object.class_variable_set(:@@x, 0)
Object.class_variable_get(:@@x)
# => 0
Object.class_variable_defined?(:@@x)
# => true

o.instance_eval { remove_instance_variable :@x }
Object.class_eval { remove_class_variable(:@@x) }
Math.send :remove_const, :PI
```

Method-объекты

```ruby
method = "string".method(:reverse)
method.call
# => "gnirts"

"hello".send :upcase
# => HELLO

"hello".public_send :upcase
# => HELLO

"hello".__send__ :upcase
# => HELLO
```

```ruby
class Square

 def area
   @side * @side
 end

 def initialize(side)
   @side = side
 end
end

area_un = Square.instance_method(:area)

s = Square.new(12)
area = area_un.bind(s)
area.call   #=> 144
```

```ruby
class Test

 def test
   :original
 end

end

um = Test.instance_method(:test)

class Test
 def test
   :modified
 end
End

t = Test.new
t.test
#=> :modified
um.bind(t).call
#=> :original
```

define_method

```ruby
class Account
 attr_accessor :state

 (1..99).each do |i|
  define_method("credit_#{i}".to_sym) do
   self.state = state + i
  end

  define_method("debit_#{i}".to_sym) do
   self.state = state - i
  end
 end

 def initialize(state)
  @state = state
 end
end

account = Account.new(20)
account.credit_31
account.state
# => 51
account.debit_45
account.state
# => 6
```

```ruby
account.public_methods(false)
# [ ... , :credit_83, :debit_83, :credit_84, :debit_84, ...]
account.respond_to?(:debit_19)
# => true
method = account.public_method(:debit_19)
# => #<Method: Account#debit_19>

class Bar
 define_method(:foo) do |arg1, arg2|
  arg1 + arg2
 end
end

a = Bar.new
a.foo
#=> ArgumentError (wrong number of arguments
(given 0, expected 2))
a.foo 1, 2
# => 3

class Bar
 define_method(:foo) do |arg=nil|
  arg
 end
end

a = Bar.new
a.foo
#=> nil
a.foo 1
# => 1
```

```ruby
class Bar
 define_method(:foo) do |*arg|
  arg
 end
end

a = Bar.new
a.foo
#=> []
a.foo 1
# => [1]
a.foo 1, 2 , 'AAA'
# => [1, 2, 'AAA']

class Bar
 define_method(:foo) do |option1: 'default
value', option2: nil|
  "#{option1} #{option2}"
 end
end

bar = Bar.new
bar.foo option2: 'Hello Yalantis !'
# => "default value Hello Yalantis !"
bar.foo option2: 'hi',option1: 'nehi'
# => "nehi hi"
```

define_singleton_method

```ruby
class Bar
 define_singleton_method(:foo) do |option1: 'default value', option2: nil|
   "#{option1} #{option2}"
 end
end

Bar.foo(option2: 'Hello Yalantis !')
# => "default value Hello Yalantis !"
```

Поиск метода в классе - возвращаемся к теме лекции 4

```ruby
class Superclass
 def action
   puts "Superclass"
 end
end
```

```ruby
module IncludedModule
 def action
   puts "Included module"
   super
 end
end
```

```ruby
module PrependedModule
 def action
   puts "Prepended module"
   super
 end
end
```

```ruby
module SingletonModule
 def action
   puts "Singleton class"
   super
 end
end
```

```ruby
class Klass < Superclass
 include IncludedModule
 prepend PrependedModule
 def action
   puts "Klass"
   super
 end
end
```

```ruby
instance = Klass.new
def instance.action
  puts "Singleton class"
  super
end
instance.action
# Singleton class
# Prepended module
# Klass
# Included module
# Superclass
```

method_missing

```ruby
class Account
 attr_accessor :state

 def initialize(state)
  @state = state
 end

 def method_missing(method_name, *args, **keyword_args, &block)
  if result = method_name.match(%r(\Adebit_\d*\z))
    self.state = state - extract_number(result)
  elsif result = method_name.match(%r(\Acredit_\d*\z))
    self.state = state + extract_number(result)
  else
    super
  end
 end

 private

 def extract_number(matched_result)
  matched_result[0].split('_')[1].to_i
 end
end
```

```ruby
account = Account.new(20)
account.debit_12
account.state
# => 8
account.credit_999
account.state
# => 1007
```

const_missing

```ruby
module Person
 def self.const_missing(name)
   puts "Oh me oh my, can't find the constant: #{name}"
 end
end

Person::JOKE
# => Oh me oh my, can't find the constant: JOKE

class Object
 def self.const_missing(name)
   puts "You can call any constants :D"
 end
end

CallMeMaybe
# You can call any constants :D
DANCE
# You can call any constants :D
```

Установка видимости метода

```
String.class_eval { private :reverse }
"hello".reverse
# => NoMethodError (private method `reverse' called for
"hello":String)

Math.sqrt(10)
# => 3.1622776601683795

Math.private_class_method(:sqrt)

Math.sqrt(10)
# NoMethodError (private method `sqrt' called for
Math:Module)

Math.public_class_method(:sqrt)

Math.sqrt(10)
# => 3.1622776601683795
```

Расширяем синглтон-класс

```ruby
module Debit
 def transaction(amount)
   self.state = state - amount
 end
end

module Credit
 def transaction(amount)
   self.state = state + amount
 end
end

class Account
 attr_accessor :state

 def initialize(state)
   @state = state
 end
end
```

```ruby
account = Account.new(20)
account.state
# => 20
account.singleton_class.include(Debit)
account.transaction(4)
account.state
# => 16

account.singleton_class.include(Credit)
account.transaction(5)
account.state
# => 21
```

Include/Extend в классе

```ruby
module Debit
 module ClassMethods
  def is_awesome?
    "is awesome"
  end
 end

 def self.included(base)
  base.extend(ClassMethods)
 end

 def transaction(amount)
  self.state = state - amount
 end
end

class Account
 include Debit

 attr_accessor :state

 def initialize(state)
  @state = state
 end
end
```

```ruby
Account.is_awesome?
# => "is Awesome"
account = Account.new(20)
account.state
# => 20
account.transaction(4)
account.state
# => 16
```

Include/Extend - расширяем сами себя

```ruby
module Bar
 def bar
   "bar"
 end
end

class Foo
 extend Bar
end

Foo.bar
 # => "bar"
```

```ruby
module Bar
 extend self

 def bar
   "bar"
 end
end

class Foo
 include Bar
end

Bar.bar
# => "bar"
Foo.new.bar
# => "bar"
```

Rails

```ruby
# rails/activesupport/lib/active_support/string_inquirer.rb
module ActiveSupport
 # Wrapping a string in this class gives you a prettier way to test
 # for equality. The value returned by <tt>Rails.env</tt> is wrapped
 # in a StringInquirer object, so instead of calling this:
 #
 #   Rails.env == 'production'
 #
 # you can call this:
 #
 #   Rails.env.production?
 class StringInquirer < String
   private

   def respond_to_missing?(method_name, include_private = false)
     method_name[-1] == '?'
   end

   def method_missing(method_name, *arguments)
     if method_name[-1] == '?'
       self == method_name[0..-2]
     else
       super
     end
   end
 end
end

Rails.env.production?
# => false
```

Sinatra Delegation

```ruby
# /lib/sinatra/base.rb
# Sinatra delegation mixin. Mixing this module into an object causes all
# methods to be delegated to the Sinatra::Application class. Used primarily
# at the top-level.
module Delegator #:nodoc:
 def self.delegate(*methods)
  methods.each do |method_name|
   define_method(method_name) do |*args, &block|
    return super(*args, &block) if respond_to? method_name
    Delegator.target.send(method_name, *args, &block)
   end
   private method_name
  end
end

 delegate :get, :patch, :put, :post, :delete, :head, :options, :link, :unlink,
          :template, :layout, :before, :after, :error, :not_found, :configure,
          :set, :mime_type, :enable, :disable, :use, :development?, :test?,
          :production?, :helpers, :settings, :register

 class << self
  attr_accessor :target
 end

 self.target = Application
end
```

```ruby
delegate :get, :patch, :put, :post, :delete
```

Paperclip

```ruby
# @name  => Имя для прикрепленного файла
# @klass => Модель ActiveRecord где данный метод будет объявлен
def define_instance_getter
 name    = @name
 options = @options
 @klass.send :define_method, @name do |*args|
   ivar = "@attachment_#{name}"
   attachment = instance_variable_get(ivar)
   if attachment.nil?
     attachment = Attachment.new(name, self, options)
     instance_variable_set(ivar, attachment)
   end
 end
end

class User
 has_attached_file :avatar, :styles => { :normal => "100x100#" }
end
```

Что почитать ?

1. https://www.tutorialspoint.com/ruby/ruby_date_time.htm
2. https://www.toptal.com/ruby/ruby-dsl-metaprogramming-guide
3. https://habr.com/ru/post/143483/
4. Глава 11 книги “Язык программирования Ruby”

Yalantis

Спасибо!

Остались вопросы? Буду рад вам ответить. Не забывайте пользоваться учебным чатом