

# Применение техник «Выделение вычислений» и «Улучшение структуры действий»

Выполнила: Емельянова Ольга ИВТ-11МО

# Изначальное состояние кода

Что показывает  
приложение:

```
# frozen_string_literal: true

module ProxyApp
  class HelpAction < GenericAction
    def perform
      puts 'Usage information:'
      puts 'list_users - list all users'
      puts 'list_user_packages <user_id> - list packages belonging to a user'
      puts 'show_package_info <proxy_package_id> - display info about package'
      puts 'list_servers - show a list of servers'
      puts 'list_server_ips <server_id> - show ips belonging to a server'
      puts 'show_ip_usage - list ips and user counts on each ip'
      puts 'show_unused_ips - list unused ips'
      puts 'show_income - list income from each user and total'
    end
  end
end
```

# Изначальное состояние кода

Модель **server** имеет one-to-many proxy\_ips,  
поля server: id, name, main\_ip

Модель **ProxyIp** имеет поля id, ip, port, country,  
server\_id, связана отношением many-to-one с  
server, many to many с proxy\_package через  
модель **ProxyIpProxyPackage**

Модель **user** имеет поля id, name, email,  
связана с **ProxyPackage** отношением  
one-to-many

| server  |
|---------|
| id      |
| name    |
| main_ip |

| ProxyIp   |
|-----------|
| id        |
| ip        |
| port      |
| country   |
| server_id |

| user  |
|-------|
| id    |
| name  |
| email |

# Описание результата применения техник к коду

// до

Первым шагом уберем  
посторонний код из файлов в  
model.rb и string\_helper.rb

```
3 module Proxyapp
4   class ProxyIp
5     attr_reader :id, :ip, :port, :country, :server_id
6
7     private_class_method :new
8
9     def self.load_data
10      data = JSON.parse(File.read(File.join(__dir__, 'proxy_ip.json')))
11      @proxy_ips = {}
12
13      for item in data
14        | @proxy_ips[item['id']] = new(item)
15      end
16    end
17
18    def initialize(data)
19      for key in data.keys
20        | instance_variable_set("@#{key}", data[key])
21      end
22    end
23
24    def self.find(id)
25      | @proxy_ips[id]
26    end
27
28    def self.where(restrictions = {})
29      result = []
30      for id in @proxy_ips.keys
31        proxy_ip = @proxy_ips[id]
32        ok = true
33        for restriction in restrictions.keys
34          | ok = false and break if proxy_ip.send(restriction) != restrictions[restriction]
35        end
36        result << proxy_ip if ok
37      end
38    end
39  end
40 end
```

// после

proxy\_ip.rb

```
1 # frozen_string_literal: true
2
3 module ProxyApp
4   class ProxyIp < Model
5     attr_reader :id, :ip, :port, :country, :server_id
6
7     belongs_to :server, :server_id
8
9     def proxy_packages
10       ProxyIpProxyPackage.get_proxy_package_for_proxy_ip_id(id)
11     end
12   end
13 end
14
```

helpers

string\_helper.rb

```
1 # frozen_string_literal: true
2
3 module ProxyApp
4   class StringHelper
5     def self.underscore(str)
6       str = str.split(':').last if str.include?(':')
7       str.gsub(/([A-Z])/) { |r| "_#{r.downcase}" }[1..]
8     end
9
10    def self.constantize(str)
11      str = str.to_s.split(':').last
12      const_name = str.to_s.gsub(/_[a-z]/) { |x| x[1..].upcase }
13      const_name[0] = const_name[0].upcase
14      ProxyApp.const_get(const_name)
15    end
16  end
17 end
18
```

// до

```
1 # frozen_string_literal: true
2
3 module ProxyApp
4   class Server
5     attr_reader :id, :name, :main_ip
6
7     private_class_method :new
8
9     def self.load_data
10      data = JSON.parse(File.read(File.join(__dir__, 'server.json')))
11      @servers = {}
12
13      for item in data
14        @servers[item['id']] = new(item)
15      end
16    end
17
18    def initialize(data)
19      for key in data.keys
20        instance_variable_set("@#{key}", data[key])
21      end
22    end
23
24    def self.find(id)
25      @servers[id]
26    end
27
28    def self.where(restrictions = {})
29      result = []
30      for id in @servers.keys
31        proxy_ip = @servers[id]
32        ok = true
33        for restriction in restrictions.keys
34          ok = false and break if proxy_ip.send(restriction) != restrictions[restriction]
35        end
36        result << proxy_ip if ok
37      end
38    end
39  end
40 end
```

// после

```
1 # frozen_string_literal: true
2
3 module ProxyApp
4   class Server < Model
5     attr_reader :id, :name, :main_ip
6
7     has_many :proxy_ip, :server_id
8   end
9 end
10
```

Все действия перенесем в отдельные файлы  processor.rb

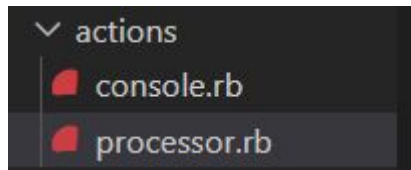
// до

```
1 module ProxyApp
2   class Processor
3     def initialize(console)
4       @console = console
5     end
6
7     def process
8       @console.print_title
9       @console.run do |line|
10         parts = line.split(/\s+/)
11         case parts[0]
12         when 'help'
13           puts 'Usage information:'
14           puts 'list_users - list all users'
15           puts 'list_user_packages <user_id> - list packages belonging to a user'
16           puts 'show_package_info <proxy_package_id> - display info about package'
17           puts 'list_servers - show a list of servers'
18           puts 'list_server_ips <server_id> - show ips belonging to a server'
19           puts 'show_ip_usage - list ips and user counts on each ip'
20           puts 'show_unused_ips - list unused ips'
21           puts 'show_income - list income from each user and total'
22         when 'list_users'
23           list_users
24         when 'list_user_packages'
25           if /\d+/ !~ parts[1].to_s
26             puts 'Wrong arguments provided, please try again'
27           else
28             list_user_packages(parts[1].to_i)
29           end
30         when 'show_package_info'
31           if /\d+/ !~ parts[1].to_s
32             puts 'Wrong arguments provided, please try again'
33           else
34             show_package_info(parts[1].to_i)
35           end
36         when 'list_servers'
37           list_servers
38         when 'list_server_ips'
39           if /\d+/ !~ parts[1].to_s
40             puts 'Wrong arguments provided, please try again'
41           else
42             list_server_ips(parts[1].to_i)
43           end
44         when 'show_ip_usage'
45           show_ip_usage
46         when 'show_unused_ips'
47           show_unused_ips
48         when 'show_income'
49           show_income
50         when 'show_ip_income'
51           show_ip_income
52       end
53     end
54   end
55 end
```

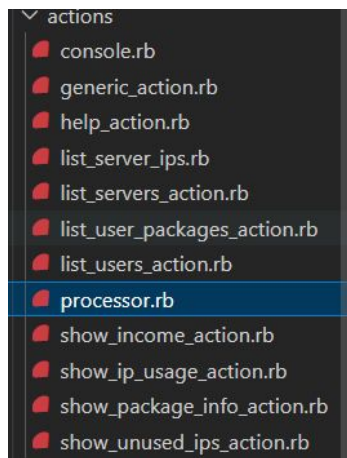
// после

```
1 # frozen_string_literal: true
2
3 module ProxyApp
4   class Processor
5     COMMANDS = {
6       'help' => [HelpAction, ProxyApp::NullValidator],
7       'list_users' => [ListUsersAction, ProxyApp::NullValidator],
8       'list_user_packages' => [ListUserPackagesAction, ProxyApp::UserIDValidator],
9       'show_package_info' => [ShowPackageInfoAction, ProxyApp::PackageIDValidator],
10      'list_servers' => [ListServersAction, ProxyApp::NullValidator],
11      'list_server_ips' => [ListServerIps, ProxyApp::ServerIDValidator],
12      'show_ip_usage' => [ShowIpUsageAction, ProxyApp::NullValidator],
13      'show_unused_ips' => [ShowUnusedIpsAction, ProxyApp::NullValidator],
14      'show_income' => [ShowIncomeAction, ProxyApp::NullValidator]
15    }.freeze
16
17     def initialize(console)
18       @console = console
19     end
20
21     def process
22       @console.print_title
23       @console.run do |line|
24         parts = line.split(/\s+/)
25         res = check_validity(line)
26         exit if parts[0] == 'exit'
27         if COMMANDS.keys.include?(parts[0]) && res[0]
28           args = parts[1..].map(&:to_i)
29           COMMANDS[parts[0]][0].new.perform(*args)
30         else
31           puts res[1]
32         end
33       end
34     end
35
36     private
37
38     def check_validity(params)
39       parts = params.split(/\s+/)
40       cmd = COMMANDS[parts[0]]
41       return [false, "Unknown command #{parts[0]}, please try again"] unless cmd
42       cmd[1].validate(parts[1..])
43     end
44
45     end
46   end
47 end
```

Структура до:



после:



show\_ip\_usage\_action.rb

```
1 module ProxyApp
2   class ShowIpUsageAction < GenericAction
3     def perform
4       ips = ProxyIp.all
5       ips.sort_by { |ip| -ip.proxy_packages.count }.each do |ip|
6         cnt = ip.proxy_packages.count
7         next if cnt.zero?
8
9         puts "#{ip.ip}:#{ip.port} - #{cnt} users"
10      end
11    end
12  end
13 end
```

list\_servers\_action.rb

```
1 module ProxyApp
2   class ListServersAction < GenericAction
3     def perform
4       servers = Server.all
5       servers.each do |server|
6         ip_count = server.proxy_ips.count
7         puts "id=#{server.id} name=#{server.name} main_ip=#{server.main_ip} ip_count=#{ip_count}"
8       end
9     end
10  end
11 end
12
```



Добавим условие выхода  processor.rb

// до

```
def process
  @console.print_title
  @console.run do |line|
    parts = line.split(/\s+/)
    res = check_validity(line)
    if COMMANDS.keys.include?(parts[0]) && res[0]
      args = parts[1..].map(&:to_i)
      COMMANDS[parts[0]][0].new.perform(*args)
    else
      puts res[1]
    end
  end
end
```

// после

```
def process
  @console.print_title
  @console.run do |line|
    parts = line.split(/\s+/)
    res = check_validity(line)
    exit if parts[0] == exit
    if COMMANDS.keys.include?(parts[0]) && res[0]
      args = parts[1..].map(&:to_i)
      COMMANDS[parts[0]][0].new.perform(*args)
    else
      puts res[1]
    end
  end
end
```

Используем рубокоп:

// до

```
def process
  @console.print_title
  @console.run do |line|
    parts = line.split(/\s+/)
    res = check_validity(line)
    exit if parts[0] == 'exit'
    if COMMANDS.keys.include?(parts[0]) && res[0]
      args = parts[1..].map(&:to_i)
      COMMANDS[parts[0]][0].new.perform(*args)
    else
      puts res[1]
    end
  end
end
```

// после

```
def process
  @console.print_title
  @console.run do |line|
    parts = line.split(/\s+/)
    res = check_validity(line)
    exit if parts[0] == 'exit'
    if COMMANDS.keys.include?(parts[0]) && res[0]
      args = parts[1..].map(&:to_i)
      COMMANDS[parts[0]][0].new.perform(*args)
    else
      puts res[1]
    end
  end
end
```

// до

```
def check_validity(params)
  parts = params.split(/\s+/)
  cmd = COMMANDS[parts[0]]
  return "Unknown command #{parts[0]}, please try again" unless cmd
```


// после

```
def check_validity(params)
  parts = params.split(/\s+/)
  cmd = COMMANDS[parts[0]]
  return [false, "Unknown command #{parts[0]}, please try again"] unless cmd

  cmd[1].validate(parts[1..])
end
```

Произведем некоторые улучшения:


// до

 list\_server\_ips.rb

```
module ProxyApp
  class ListServerIps < GenericAction
    def perform(server_id)
      server = Server.find(server_id)
      unless server
        puts "No such server with id #{server_id}"
        return
      end
      ips = server.proxy_ips
      ips.each do |ip|
        puts "#{ip.ip}:#{ip.port}"
      end
      puts "#{ips.count} IPs total"
    end
  end
end
```

// после

```
module ProxyApp
  class ListServerIps < GenericAction
    def perform(server_id)
      server = Server.find(server_id)
      ips = server.proxy_ips
      ips.each do |ip|
        puts "#{ip.ip}:#{ip.port}"
      end
      puts "#{ips.count} IPs total"
    end
  end
end
```


 show\_ip\_usage\_action.rb

// до

```
ips.sort_by { |ip| -ip.proxy_packages.count }.each do |ip|
  cnt = ip.proxy_packages.count
  next if cnt == 0
```

// после

```
ips = ProxyIp.all
ips.sort_by { |ip| -ip.proxy_packages.count }.each do |ip|
  cnt = ip.proxy_packages.count
  next if cnt.zero?
```

 show\_unused\_ips\_action.rb

// до

```
ips.each do |ip|
  cnt = ip.proxy_packages.count
  next if cnt > 0
```

// после

```
ips.each do |ip|
  cnt = ip.proxy_packages.count
  next if cnt.postive?
```

Произведем улучшения  proxy\_ip\_proxy\_package.rb

// до

```
for item in data
  @proxy_ip_to_proxy_packages[item['proxy_ip_id']] << item['proxy_package_id']
  @proxy_package_to_proxy_ips[item['proxy_package_id']] << item['proxy_ip_id']
end
```

// после

```
data.each do |item|
  @proxy_ip_to_proxy_packages[item['proxy_ip_id']] << item['proxy_package_id']
  @proxy_package_to_proxy_ips[item['proxy_package_id']] << item['proxy_ip_id']
end
```

// до

```
def self.get_proxy_ip_for_proxy_package_id(id)
  result = []
  for pid in @proxy_package_to_proxy_ips[id]
    result << ProxyIp.find(pid)
  end
  result
end
```

// после

```
def self.get_proxy_ip_for_proxy_package_id(id)
  @proxy_package_to_proxy_ips[id].map { |pid| ProxyIp.find(pid) }
end
```

// до

```
def self.get_proxy_package_for_proxy_ip_id(id)
  result = []
  for pid in @proxy_ip_to_proxy_packages[id]
    result << ProxyPackage.find(pid)
  end
  result
end
```

// после

```
def self.get_proxy_package_for_proxy_ip_id(id)
  @proxy_ip_to_proxy_packages[id].map { |pid| ProxyPackage.find(pid) }
end
```

Спасибо за внимание!