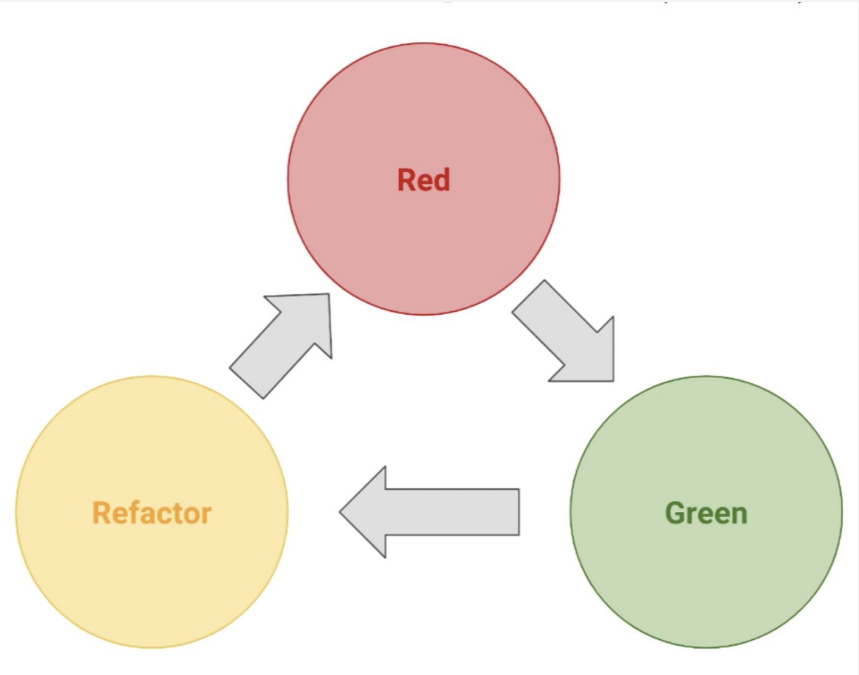
Yalantis

Лекция 16: Тестирование Rails

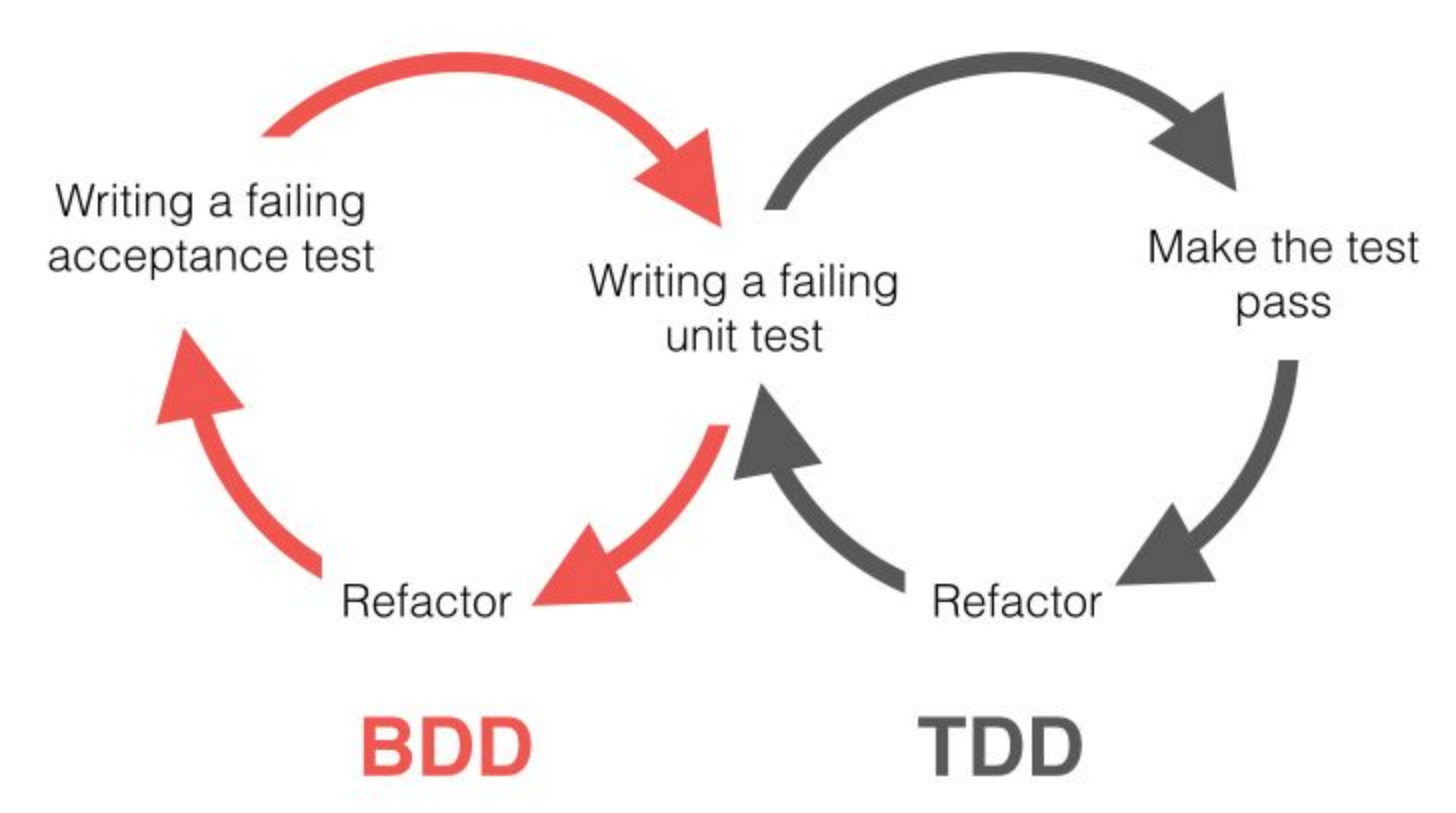
Курс лекций по основами web-разработки на языке программирования Ruby

Yalantis

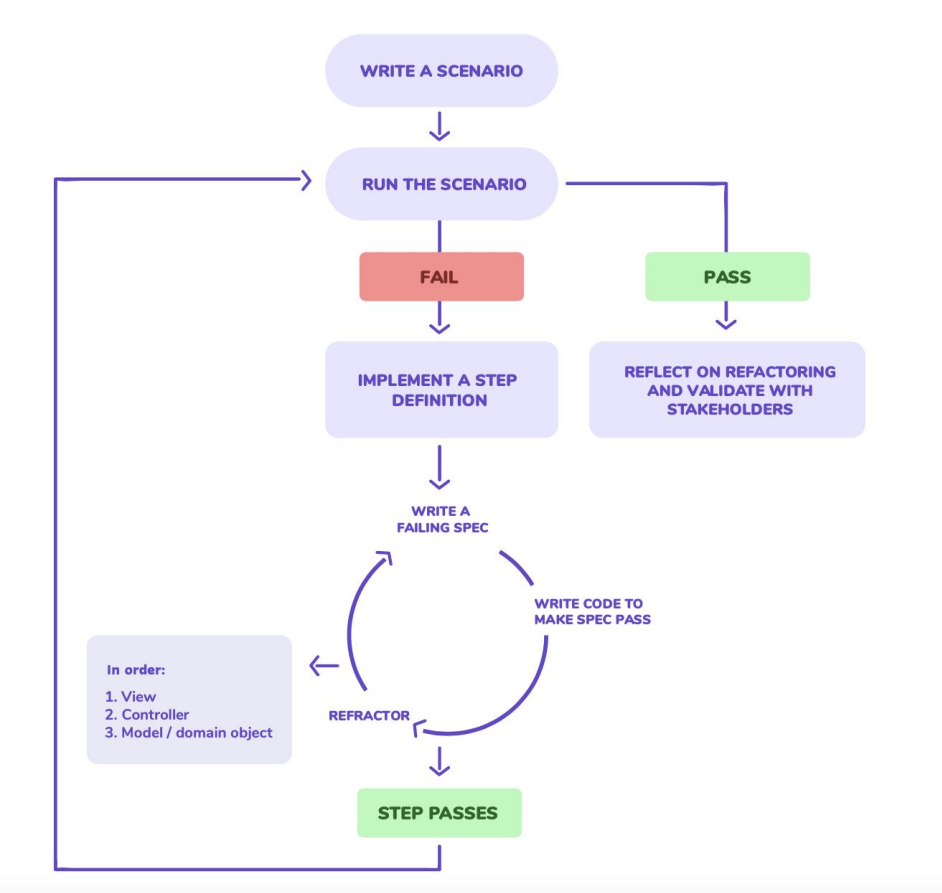
Test Driven Development(TDD)



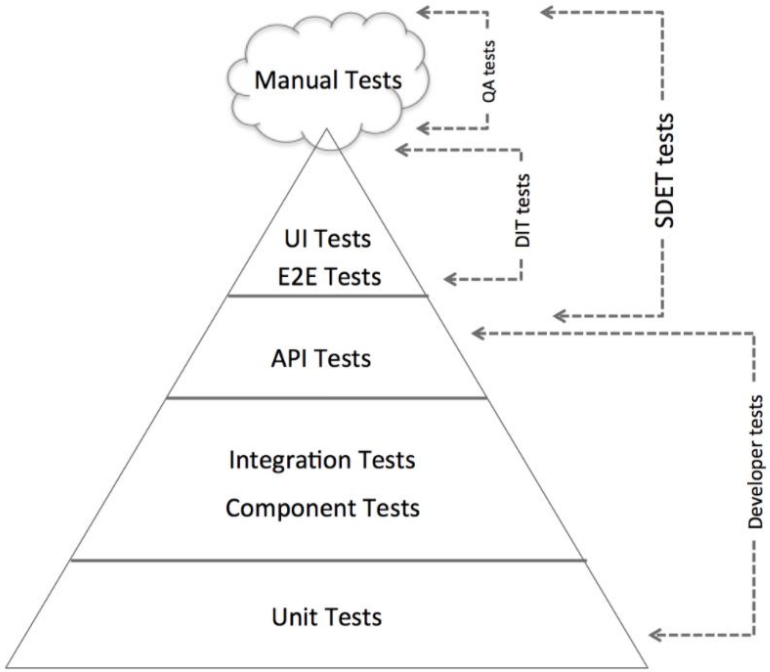
Yalantis


Writing a failing acceptance test
Writing a failing unit test
Make the test pass
Refactor
Refactor
BDD
TDD

Yalantis


WRITE A SCENARIO
RUN THE SCENARIO
FAIL
PASS
IMPLEMENT A STEP DEFINITION
REFLECT ON REFACTORING AND VALIDATE WITH STAKEHOLDERS
WRITE A FAILING SPEC
WRITE CODE TO MAKE SPEC PASS
REFACTOR
In order:
1. View
2. Controller
3. Model / domain object
STEP PASSES

Yalantis


Manual Tests
QA tests
DIT tests
SDET tests
UI Tests
E2E Tests
API Tests
Integration Tests
Component Tests
Unit Tests
Developer tests

Yalantis

Unit

```ruby
RSpec.describe 'change matcher' do
 subject { [1, 2, 3, 4] }

 it 'checks that a method changes object state' do
   expect { subject.push(4) }.to change { subject.length }.by(1)
 end

 it 'accepts negative arguments' do
   expect { subject.pop }.to change { subject.length }.by(-1)
 end
end
```

Feature

```ruby
#spec/features/add_task_spec.rb
RSpec.describe "adding a new task" do
 let!(:project) { create(:project, name: "Project Bluebook") }
 let!(:task_1) { create(:task, project: project, title: "Search Sky", size:
1) }
 let!(:task_2) { create(:task, project: project, title: "Use Telescope",
size: 1) }
 it "can add a task" do
  visit(project_path(project))
   fill_in("Task", with: "Find UFOs")
   select("2", from: "Size")
   click_on("Add Task")
   expect(current_path).to eq(project_path(project))
   within("#task_3") do
     expect(page).to have_selector(".name", text: "Find UFOs")
     expect(page).to have_selector(".size", text: "2")
     expect(page).not_to have_selector("a", text: "Down")
   end
 end
end
```

Yalantis

```
<h3>New Task</h3>
<%= form_for Task.new(project_id: @project.id) do |f| %>
  <%= f.hidden_field :project_id %>
  <%= f.label :title, "Task" %>
  <%= f.text_field :title %>
  <%= f.label :size %>
  <%= f.select :size, [1, 2, 3, 4, 5] %>
  <%= f.submit "Add Task" %>
<% end %>

<h2>Project <%= @project.name %></h2>
<h3>Existing Tasks:</h3>
<table>
   <thead>
   <tr>Name</tr>
   <tr>Size</tr>
   </thead>
   <tbody>
   <% @project.tasks.each do |task| %> <tr>
      <td class="name"><%= task.title %></td>
      <td class="size"><%= task.size %></td>
      <td class="completed"><%= task.completed_at %></td>
   </tr>
   <% end %>
   </tbody>
</table>
```

Cucumber

Yalantis

```
#integrationfeatures/add_task.feature
Feature: Adding a task
Background:
  Given a project
Scenario: I can add and change the priority of a new task
When I visit the project page
And I complete the new task form
Then I am back on the project page
And I see the new task is last in the list
When I click to move the new task up
Then I am back on the project page
And the new task is in the middle of the list
```

Cucumber

Yalantis

```ruby
#integrationfeatures/step_definitions/add_task_steps.rb
Given(/^a project$/) do
 @project = Project.create(name: "Bluebook")
 @project.tasks.create(title: "Hunt the aliens", size: 1,
project_order: 1)
 @project.tasks.create(title: "Write a book", size: 1,
project_order: 2)
end
When(/^I visit the project page$/) do
 visit project_path(@project)
end
When(/^I complete the new task form$/) do
 fill_in("Task", with: "Find UFOs")
 select("2", from: "Size")
 click_on("Add Task")
end
Then(/^I am back on the project page$/) do
 expect(current_path).to eq(project_path(@project))
end
```

```ruby
Then(/^I see the new task is last in the list$/) do
 within("#task_3") do
  expect(page).to have_selector(".name", text: "Find UFOs")
  expect(page).to have_selector(".size", text: "2")
  expect(page).not_to have_selector("a", text: "Down")
 end
end

When(/^I click to move the new task up$/) do
 within("#task_3") do
  click_on("Up")
 end
end
Then(/^the new task is in the middle of the list$/) do
 within("#task_2") do
  expect(page).to have_selector(".name", text: "Find UFOs")
 end
end
```

Yalantis

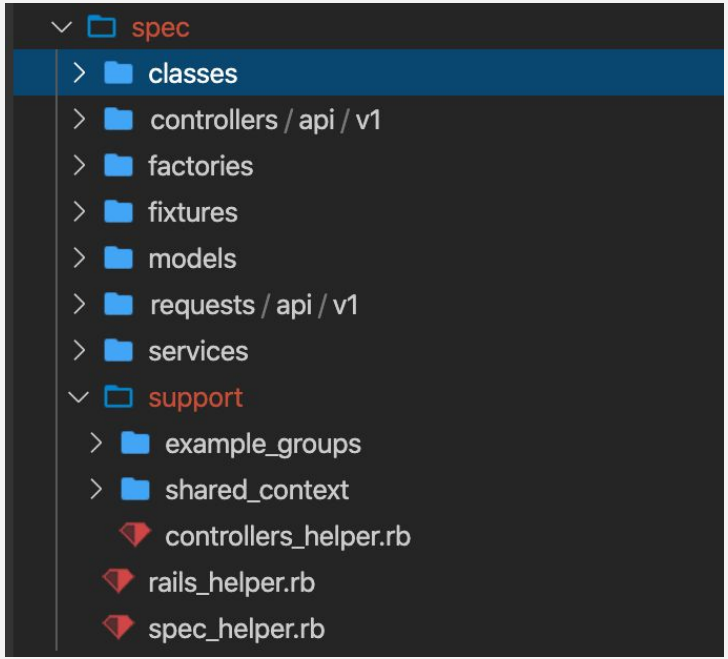
Экосистема RSpec

```
gem 'rspec-core'
#предоставляет структуру для написания примеров (test suites) и команду rspec
RSpec.describe Order do
 it "sums the prices of its line items" do
   order = Order.new
   order.add_entry(LineItem.new(:item => Item.new(
      :price => Money.new(1.11, :USD)
   )))
   order.add_entry(LineItem.new(:item => Item.new(
      :price => Money.new(2.22, :USD),
      :quantity => 2
   )))
   expect(order.total).to eq(Money.new(5.55, :USD))
 end
end

gem 'rspec-expectations'
#позволяет выразить ожидаемые результаты
expect(account.balance).to eq(Money.new(37.42, :USD)

gem 'rspec-mocks'
#используется для имитации поведения классов и объектов (моки и стабы)
book = instance_double("Book", :pages => 250)
allow(book).to receive(:title) { "The RSpec Book" }
```

Installation

Yalantis

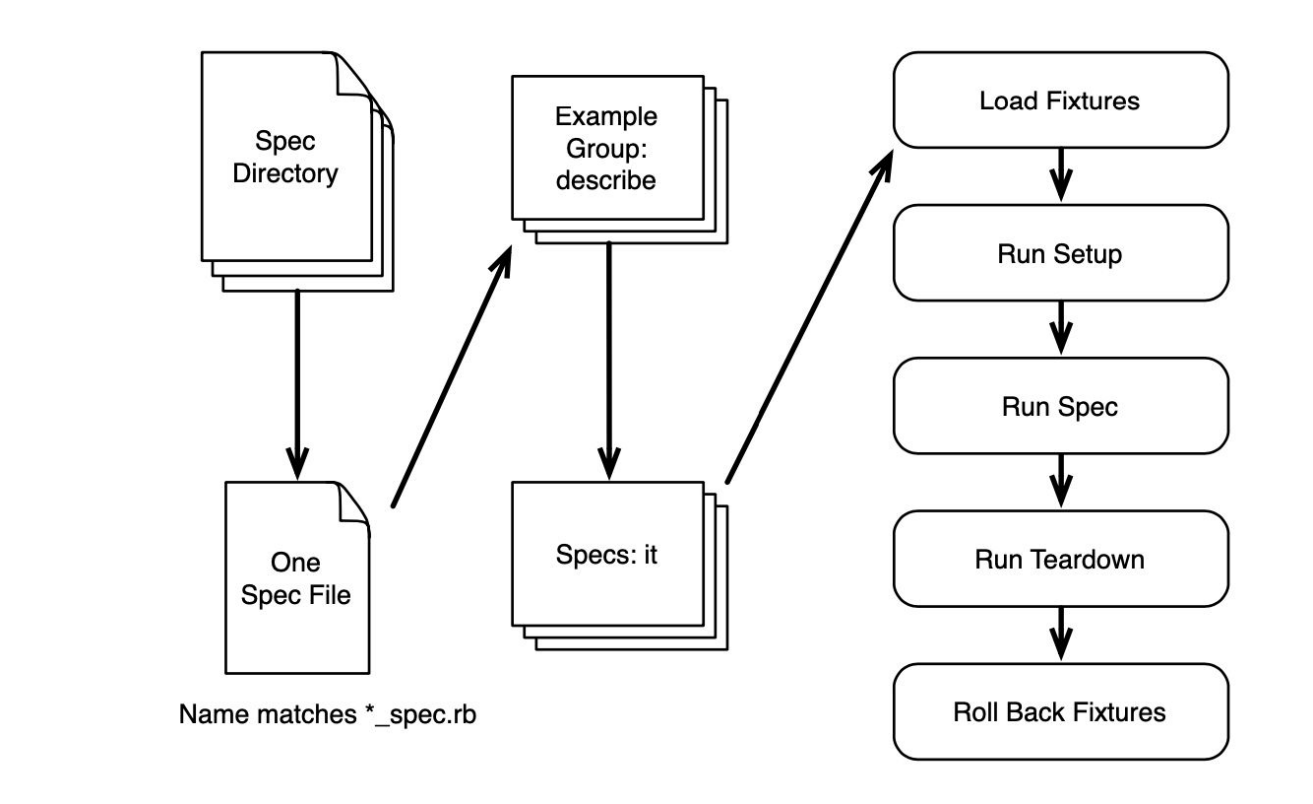
```
#Gemfile
gem 'rspec'

$ bundle install
$ rspec --init
create   .rspec
create   spec/spec_helper.rb
```




```
#.rspec

--require rails_helper
--colour
--format documentation
--order rand
```

Yalantis

Rspec spec



Yalantis

Настройка конфигурации

```
#spec/rails_helper.rb
require 'simplecov'
SimpleCov.start 'rails' do
 add_filter '/spec/'
 add_filter '/config/'
end

require 'spec_helper'
ENV['RAILS_ENV'] ||= 'test'
Dir['./spec/support/**/*.rb'].sort.each { |f| require f }
require File.expand_path('../config/environment', __dir__)
# Prevent database truncation if the environment is production
abort('The Rails environment is running in production mode!') if Rails.env.production?
require 'rspec/rails'
require 'shoulda-matchers'

begin
 ActiveRecord::Migration.maintain_test_schema!
rescue ActiveRecord::PendingMigrationError => e
 puts e.to_s.strip
 exit 1
end
```

Yalantis

Настройка конфигурации

```
#spec/rails_helper.rb
RSpec.configure do |config|
 config.fixture_path = "#{::Rails.root}/spec/fixtures"
 config.use_transactional_fixtures = true
 config.include FactoryBot::Syntax::Methods
 config.infer_spec_type_from_file_location!
 config.filter_rails_from_backtrace!

 config.before(:suite) do
   DatabaseCleaner.strategy = :transaction
   DatabaseCleaner.clean_with(:truncation)
 end
 config.around(:each) do |example|
   DatabaseCleaner.cleaning do
     example.run
   end
 end
end
Shoulda::Matchers.configure do |config|
 config.integrate do |with|
   with.test_framework :rspec
   with.library :rails
 end
```

Yalantis

Настройка конфигурации

```
app
├── controllers
│   ├── application_controller.rb
│   └── books_controller.rb
├── helpers
│   ├── application_helper.rb
│   └── books_helper.rb
├── models
│   ├── author.rb
│   └── book.rb
└── views
    ├── books
    └── layouts
lib
├── country_map.rb
├── development_mail_interceptor.rb
├── enviroment_mail_interceptor.rb
└── tasks
    └── irc.rake
spec
├── controllers
│   └── books_controller_spec.rb
├── country_map_spec.rb
├── features
│   └── tracking_book_delivery_spec.rb
├── helpers
│   └── books_helper_spec.rb
├── models
│   ├── author_spec.rb
│   └── book_spec.rb
├── rails_helper.rb
├── requests
│   └── books_spec.rb
├── routing
│   └── books_routing_spec.rb
├── spec_helper.rb
├── tasks
│   └── irc_spec.rb
└── views
    └── books
```

```ruby
RSpec.configure do |config|
 config.infer_spec_type_from_file_location!
end

RSpec.describe WidgetsController do
 it "responds successfully" do
   get :index
   expect(response.status).to eq(200)
 end
end

RSpec.describe WidgetsController, :type => :controller do
 it "responds successfully" do
   get :index
   expect(response.status).to eq(200)
 end
end
# set `:type` for serializers directory
RSpec.configure do |config|
 config.define_derived_metadata(:file_path =>
Regexp.new('/spec/serializers/')) do |metadata|
   metadata[:type] = :serializer
 end
end
```

Yalantis

Let

```
class ProgrammingLanguage
 attr_reader :name

 def initialize(name = 'Ruby')
   @name = name
 end
end

RSpec.describe ProgrammingLanguage do
 let(:language) { ProgrammingLanguage.new('Python') }

 it 'should store the name of the language' do
   expect(language.name).to eq('Python')
 end

 context 'with no argument' do
   let(:language) { ProgrammingLanguage.new }

   it 'should default to Ruby as the name' do
     expect(language.name).to eq('Ruby')
   end
 end
end
```

Yalantis

Subject

```
class Queen
 attr_reader :name

 def initialize(name)
   @name = name
 end
end

RSpec.describe Queen do
 subject { described_class.new('Helen') }
 let(:louis) { described_class.new('Louis') }

 it 'represents a great person' do
   expect(subject.name).to eq('Helen')
   expect(louis.name).to eq('Louis')
 end
end
```

```
RSpec.describe Array do
 describe "with 3 items" do
   subject { [1,2,3] }
   it { should_not be_empty }
   # or
   it { is_expected.not_to be_empty }
 end
end
```

Yalantis

RSpec’s basic matchers

```
expect(actual).to be > expected #также с <, >=, <=, и ==
expect(actual).to be_a(type)
expect(actual).to be_truthy
expect(actual).to be_falsy
expect(actual).to be_nil
expect(actual).to be_between(min, max)
expect(actual).to be_within(delta).of(expected)
expect { block }.to change(receiver, message, &block)
expect(actual).to contain_exactly(expected)
expect(range).to cover(actual_value)
expect(actual).to eq(expected)
expect(actual).to exist
expect(actual).to have_attributes(key/value pairs)
expect(actual).to include(*expected)
expect(actual).to match(regex)
expect { block }.to raise_error(exception)
expect(actual).to satisfy { block }
```

Yalantis

Write custom matcher

```
RSpec::Matchers.define :be_of_size do |expected|
 match do |actual|
   actual.size == expected
 end

 description do
   "have tasks totaling #{expected} points"
 end

 failure_message do |actual|
   "expected project #{actual.name} to have size #{expected}, was #{actual}"
 end

 failure_message_when_negated do |actual|
   "expected project #{actual.name} not to have size #{expected}, but it did"
 end
end

it "can calculate total size" do
 expect(project).to be_of_size(10)
 expect(project).not_to be_of_size(5)
end
```

Yalantis

Хуки

:each/:example - выполняется до/после каждого примера
:all/:context - выполняется до/после группы верхнего уровня
:suite - запускается один раз вначале и/или после выполнения последнего примера

```
RSpec.describe 'before and after hooks' do
 before(:context) do
   puts 'Before context'
 end

 after(:context) do
   puts 'After context'
 end

 before(:each) do
   puts 'Before example'
 end

 after(:each) do
   puts 'After example'
 end

 it 'is just a random example' do
   puts 'Run example 1'
   expect(5 * 4).to eq(20)
 end

 it 'is just another random example' do
   puts 'Run example 2'
   expect(3 - 2).to eq(1)
 end
end
```

$ rspec spec/example_spec.rb

```
before and after hooks
Before context
Before example
Run example 1
After example
  is just a random example
Before example
Run example 2
After example
  is just another random example
After context

Finished in 0.00129 seconds (files took 0.07956 seconds to load)
2 examples, 0 failures
```

Yalantis

Вложенные хуки

```ruby
RSpec.describe 'Nested hooks' do
 before(:context) do
  puts 'OUTER Before context'
 end

 before(:example) do
  puts 'OUTER Before example'
 end

 it 'does basic math' do
  expect(1 + 1).to eq(2)
 end

 context 'with condition A' do
  before(:context) do
   puts 'INNER Before context'
  end

  before(:example) do
   puts 'INNER Before example'
  end

  it 'does some more basic math' do
   expect(1 + 1).to eq(2)
  end
 end
end
```

$ rspec spec/example_spec.rb

```
Nested hooks
OUTER Before context
OUTER Before example
  does basic math
  with condition A
INNER Before context
OUTER Before example
INNER Before example
    does some more basic math

Finished in 0.00193 seconds (files took 0.1329 seconds to load)
2 examples, 0 failures
```

Yalantis

Pending Tests

```
RSpec.describe 'Rending' do

 xit 'does basic math' do
   expect(1 + 1).to eq(2)
 end

 xcontext 'with condition A' do
   before(:context) do
     puts 'INNER Before context'
   end

   it 'does some more basic math' do
     expect(1 + 1).to eq(2)
   end
   it 'does some more more basic math' do
     expect(1 + 1).to eq(2)
   end
 end
it 'does something else' do
 skip
end

it 'does something else' do
 skip 'reason explanation'
end
end
```

Yalantis

Shared examples

```
RSpec.shared_examples 'a Ruby object with three elements' do
 it 'returns the number of items' do
   expect(subject.length).to eq(3)
 end
end

RSpec.describe Array do
 subject { [1, 2, 3] }
 include_examples 'a Ruby object with three elements'
end

RSpec.describe String do
 subject { 'abc' }
 include_examples 'a Ruby object with three elements'
end

RSpec.describe Hash do
 subject {{ a: 1, b: 2, c: 3 }}
 include_examples 'a Ruby object with three elements'
end

class SausageLink
 def length
   3
 end
end

RSpec.describe SausageLink do
 subject { described_class.new }
 include_examples 'a Ruby object with three elements'
end
```

Yalantis

Shared context

```ruby
RSpec.shared_context 'common' do
 before do
  @foods = []
 end

 def some_helper_method
  5
 end

 let(:some_variable) { [1, 2, 3] }
end
```

```ruby
RSpec.describe 'first example group' do
 include_context 'common'

 it 'can use outside instance variables' do
  expect(@foods.length).to eq(0)
  @foods << 'Sushi'
  expect(@foods.length).to eq(1)
 end

 it 'can reuse instance variables across different examples' do
  expect(@foods.length).to eq(0)
 end

 it 'can use shared helper methods' do
  expect(some_helper_method).to eq(5)
 end
end

RSpec.describe 'second example in different file' do
 include_context 'common'

 it 'can use shared let variables' do
  expect(some_variable).to eq([1, 2, 3])
 end
end
```

Yalantis

Fixtures

```
runway:
 name: Project Runway
 due_date: 2016-12-18
 description: |
   The awesomest project ever.
   It's really, really great.
```

```
runway:
 name: Project Runway
 due_date: <%= 1.month.from_now %>
```

```
<% 10.times do |i| %>
task_<%=i%>:
name: "Task <%= i %>"
<% end %>
```

Yalantis

Double

```
describe NotificationsController do
 # NotificationsController загружает последние уведомления
 # со стороннего сервиса по HTTP
 # с помощью NotificationsDatasource.
 let(:datasource) do
   double(:datasource, as_json: { notifications: [] })
 end

 before do
   # Подменяем реальный NotificationsDatasource дублером,
   # чтобы не зависеть от внешнего сервиса в тестах:
   allow(NotificationsDatasource).to receive(:new).and_return(datasource)
 end

 describe "#index" do
   it "wraps notifications in 'data' key" do
     get :index, format: :json

     expect(json_response["data"].keys).to have_key "notifications"
   end
 end
end
```

Yalantis

Stub

```
context "when attachment file is too large to email" do
 let(:max_file_size) { Attachment::MAX_FILE_SIZE }

 before do
   allow(attachment)
       .to receive(:file_size)
               .and_return(max_file_size + 1)
 end

 it "raises 'file is too large' error" do
   # ...
 end
end
```

```
RSpec.describe 'null object' do
 it 'allow any method' do
   null_object = double('null object').as_null_object
   expect(null_object).to respond_to(:any_undefined_method)
 end
end
```

```
RSpec.describe 'allow_any_instance_of' do
 it 'yields the receiver to the block implementation' do
   allow_any_instance_of(String).to receive(:slice).and_return(:return_value)

   expect('string'.slice(2, 3)).to eq('rin')
 end
end
```

Yalantis

Mock

```ruby
RSpec.describe 'allow method review' do
 it 'can customize return value for methods on doubles' do
   calculator = double
   allow(calculator).to receive(:add).and_return(15)

   expect(calculator.add).to eq(15)
   expect(calculator.add(3)).to eq(15)
   expect(calculator.add(-2, -3 -5)).to eq(15)
   expect(calculator.add('hello')).to eq(15)
 end

 it 'can stub one or more methods on a real object' do
   arr = [1, 2, 3]
   allow(arr).to receive(:sum).and_return(10)
   expect(arr.sum).to eq(10)

   arr.push(4)
   expect(arr).to eq([1, 2, 3, 4])
 end
end
```

Yalantis

Factory Bot

```
RSpec.configure do |config|
 config.include FactoryBot::Syntax::Methods
end
```

```
FactoryBot.define do
 factory :project do
 name "Project Runway"
 due_date { Date.today - rand(50) }
 slug { "#{name.downcase.gsub(" ", "_")}" }
 end
end
```

```
it "uses factory_bot slug block" do
 project = FactoryBot.create(:project, name: "Book To
Write")
 expect(project.slug).to eq("book_to_write")
end
```

```
FactoryBot.define do
 factory :task do
 title "Do Something"
 size 3
 project
 end
end
```

```
FactoryBot.define do
 factory :task do
  title "To Something"
  size 3
  project
  association :doer, factory: :user, strategy: :build
 end
end
```

Yalantis

Factory Bot

- build (:project) - возвращает экземпляр модели, который не был сохранен в базе данных.
- create (:project) - возвращает экземпляр модели и сохраняет его в базе данных.
- attribute_for (:project) - возвращает хэш всех атрибутов фабрики, которые подходят для передачи в ActiveRecord # new или ActiveRecord # create.
- build_stubbed (: project). Как и build, он возвращает несохраненный объект модели. Он присваивает модели фиктивный идентификатор ActiveRecord и отключает методы взаимодействия с базой данных, так что тест вызывает исключение, если они вызываются.

Yalantis

Shoulda matchers.FFaker

```
RSpec.describe MenuItem, type: :model do
 describe 'associations' do
   it { should belong_to(:category).class_name('MenuCategory') }
 end

 describe 'validations' do
   it { should validate_presence_of(:name) }
   it { should validate_uniqueness_of(:name).scoped_to(:category_id) }
 end
end
```

Yalantis

Спасибо!

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