daftcode

RUBY ON RAILS
Level UP

5. Testy

Praca domowa

RUBY ON RAILS
Level UP

- Wstęp
- RSpec
- Unit tests
- TDD
- Model specs
- Integration tests
- Pomocne narzędzia

RUBY ON RAILS
Level UP

Po co pisać testy?

RUBY ON RAILS
Level UP

RSpec

```
# Gemfile

gem 'rspec-rails'

$ rails generate rspec:install
```

```
create  .rspec
create  spec
create  spec/spec_helper.rb
create  spec/rails_helper.rb
```

RUBY ON RAILS
Level UP

Przykładowy test

```ruby
# spec/services/calculator_spec.rb

RSpec.describe Calculator, type: :service do
  describe '#sum' do
    it 'adds two and two' do
      expect(Calculator.new(first_number: 2, second_number: 2).sum).to eq(4)
    end
  end
end
```

```
➜  homework git:(tests) ✗ rspec spec/services/calculator_spec.rb
.

Finished in 0.01458 seconds (files took 6.61 seconds to load)
1 example, 0 failures
```

Four Phase Testing

- setup
- exercise
- verification
- teardown

Deterministic & Independent

Przykładowy test

```
➜  homework git:(tests) ✗ rspec spec/services/calculator_spec.rb
F

Failures:

  1) Calculator#sum adds two and two
     Failure/Error: expect(Calculator.new(first_number: 2, second_number: 2).sum).to eq(4)

       expected: 4
            got: 1010

       (compared using ==)
     # ./spec/services/calculator_spec.rb:6:in `block (3 levels) in <top (required)>'

Finished in 0.00746 seconds (files took 1.23 seconds to load)
1 example, 1 failure
```

RSpec

- describe
- it
- expect
- context
- let
- subject
- before, after, around

describe

```ruby
RSpec.describe Calculator do
  describe '#sum' do
    # ...
  end

  describe '.max_number' do
    # ...
  end
end
```

it

```ruby
RSpec.describe Calculator do
  describe '#sum' do
    it 'adds two and two' do
      # ...
    end
  end
end
```

expect

```ruby
RSpec.describe Calculator do
  describe '#sum' do
    it 'adds two and two' do
      expect(Calculator.new(2, 2).sum).to eq(4)
    end
  end
end
```

matchers

```ruby
expect([1,2,3]).to include(2)

expect(:some_symbol).to be_a(Symbol)

expect(true).not_to be_falsey

expect { raise 'Ooopsie' }.to raise_error(StandardError)

expect { SamuraiKiller.new(samurai).call }.to change { Samurai.count }.by(-1)

expect(response).to have_http_status(200)
```

subject

```ruby
RSpec.describe Calculator do
  subject(:calculator) { Calculator.new(2, 2) }

  describe '#sum' do
    it 'returns 4' do
      expect(calculator.sum).to eq(4)
    end

    it 'does not return 5' do
      expect(calculator.sum).not_to eq(5)
    end
  end
end
```

let

```ruby
RSpec.describe Calculator do
  subject(:calculator) { Calculator.new(params) }

  describe '#sum' do
    let(:params) { { first_number: first_number, second_number: 2 } }
    let(:first_number) { 2 }

    it 'returns 4' do
      expect(calculator.sum).to eq(4)
    end
  end
end
```

context

```ruby
RSpec.describe Calculator do
  subject(:calculator) { Calculator.new(params) }

  describe '#sum' do
    context 'with given 2 and 2' do
      let(:params) { { first_number: 2, second_number: 2 } }

      it 'returns 4' do
        expect(calculator.sum).to eq(4)
      end
    end

    context ‘with given 3 and 3' do
      let(:params) { { first_number: 3, second_number: 3 } }

      it 'returns 6' do
        expect(calculator.sum).to eq(6)
      end
    end
  end
end
```

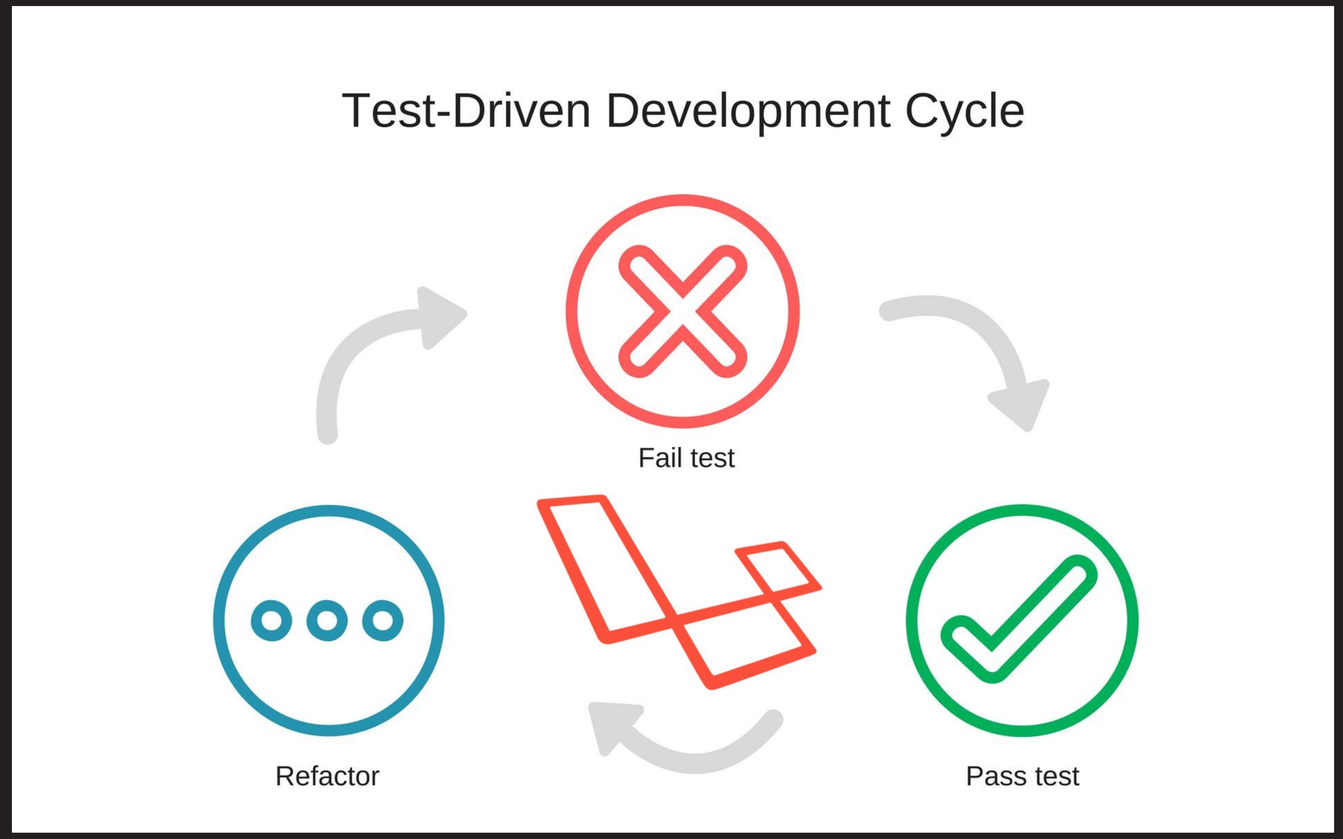
before

```
context 'when signed in' do
  before { sign_in(user) }

  it 'responds with 200' do
    # ...
  end
end

context 'when signed out' do
  before { sign_out(user) }

  it 'responds with 401' do
    # ...
  end
end
```

Unit tests
+ TDD

RUBY ON RAILS
Level UP

Test Driven Development



RUBY ON RAILS
Level UP

Przykład - Reverser

```ruby
describe Reverser do
  subject(:reverser) { described_class.new(params) }

  let(:params) { { sentence: sentence } }

  describe '#call' do
    context 'with basic sentence' do
      let(:sentence) { 'ala ma kota' }

      it 'reverses sentece' do
        expect(reverser.call).to eq('kota ma ala')
      end
    end
  end
end
```

Przykład - Reverser

```
An error occurred while loading ./spec/services/reverser_spec.rb.
Failure/Error:
  describe Reverser do
    subject(:reverser) { described_class.new(params) }

    let(:params) { { sentence: sentence } }

    describe '#call' do
      context 'with basic sentence' do
        let(:sentence) { 'ala ma kota' }

        it 'reverses sentece' do

NameError:
  uninitialized constant Reverser
```

Przykład - Reverser

```ruby
class Reverser
  def initialize(sentence:)
    @sentence = sentence
  end

  def call
    'kota ma ala'
  end
end
```

```
Finished in 0.0051 seconds (files took 1.36 seconds to load)
1 example, 0 failures
```

Przykład - Reverser

```ruby
describe Reverser do
  subject(:reverser) { described_class.new(params) }

  let(:params) { { sentence: sentence } }

  describe '#call' do
    context 'with basic sentence' do
      let(:sentence) { 'ala ma kota' }

      it 'reverses sentece' do
        expect(reverser.call).to eq('kota ma ala')
      end
    end

    context 'with empty string' do
      let(:sentence) { '' }

      it 'returns empty string' do
        expect(reverser.call).to eq('')
      end
    end
  end
end
```

Przykład - Reverser

```ruby
class Reverser
  def initialize(sentence:)
    @sentence = sentence
  end

  def call
    sentence.split(separator).reverse.join(separator)
  end

  private

  attr_reader :sentence, :censor

  def separator
    ' '
  end
end
```

```
..

Finished in 0.00573 seconds (files took 1.2 seconds to load)
2 examples, 0 failures
```

Stubs

```ruby
class Reverser
  def initialize(sentence:, censor: CobolCensor)
    @sentence = sentence
    @censor = censor
  end

  def call
    censored_sentence.split(separator).reverse.join(separator)
  end

  private

  attr_reader :sentence, :censor

  def censored_sentence
    censor.call(sentence)
  end

  def separator
    ' '
  end
end
```

Stubs

```ruby
describe Reverser do
  subject(:reverser) { described_class.new(params) }

  let(:params) { { sentence: sentence, censor: censor_double } }
  let(:censor_double) { class_double('Censor') }

  before { allow(censor_double).to receive(:call).with(sentence).and_return(sentence) }

  describe '#call' do
    context 'with basic sentence' do
      # ...
    end

    context 'with empty string' do
      # ...
    end
  end
end
```

Model specs

RUBY ON RAILS
Level UP

factory_bot

```ruby
# spec/factories/clans.rb

FactoryBot.define do
  factory :clan do
    sequence(:name) { |i| "Clan_#{i}" }
  end
end
```

```ruby
# spec/factories/warriors.rb

FactoryBot.define do
  factory :warrior do
    association(:clan)
    name { 'Gorn' }

    trait :experienced do
      number_of_battles { 100 }
    end
  end
end
```

```ruby
FactoryBot.create(:warrior)
FactoryBot.create(:warrior, name: 'Cor Angar')
FactoryBot.create(:warrior, :experienced)
```

Przykładowy test modelu

```ruby
# spec/models/clan_spec.rb

describe Clan, type: :model do
  subject(:clan) { build(:clan) }

  it 'is valid' do
    expect(clan).to be_valid
  end

  context 'without name' do
    before { clan.name = nil }

    it 'is not valid' do
      expect(clan).to be_invalid
    end
  end
end
```

Integration tests

Request specs

RUBY ON RAILS
Level UP

Przykład - GET

```ruby
# spec/requests/clans.rb

describe 'Clans API' do
  describe 'GET /clans' do
    it 'responds with 200' do
      get '/clans'
      expect(response).to have_http_status(200)
    end
  end
end
```

Przykład - GET

```
describe 'Clans API' do
  describe 'GET /clans' do
    before { create_list(:clan, 2) }

    it ‘includes correct number of records’ do
      get "/clans"
      response_json = JSON.parse(response.body)
      expect(response_json['data'].size).to eq(2)
    end
  end
end
```

Przykład - GET

```ruby
describe 'GET /clans/:id' do
  let(:clan) { create(:clan, name: clan_name) }
  let(:clan_name) { 'Nowy Oboz' }

  it ‘includes correct name’ do
    get "/clans/#{clan.id}"
    response_json = JSON.parse(response.body)
    expect(response_json.dig('data', 'attributes', 'name')).to eq(clan_name)
  end
end
```

Przykład - POST

```
describe 'POST /clans' do
  before { post '/clans', params: params }

  context 'with valid name' do
    let(:params) { { name: 'Stary Oboz’ } }

    it 'responds with 201' do
      expect(response.status).to eq(201)
    end
  end
end
```

Przykład - POST

```
context 'with invalid name' do
  let(:params) { { name: 'a' } }

  it 'responds with 422' do
    post '/clans', params: params
    expect(response.status).to eq(422)
  end

  it 'does not create clan' do
    expect { post '/clans' }.not_to change { Clan.count }
  end
end
```

shared examples

```ruby
shared_examples_for 'unprocessable entity' do
  it 'responds with 422' do
    expect(response.status).to eq(422)
  end
end
```

```ruby
describe 'POST /clans' do
  before { post '/clans', params: params }

  context 'with invalid params' do
    context 'with too short name' do
      let(:params) { { name: 'a' } }

      include_examples 'unprocessable entity'
    end

    context 'with blank name' do
      let(:params) { { name: nil } }

      include_examples 'unprocessable entity'
    end
  end
end
```

Przydatne narzędzia

RUBY ON RAILS
Level UP

Coverage

Coverage

```
gem 'simplecov', group: :test
```


```
gem 'deep-cover', group: :test
```

```
raise 'Ooopsie' if some_condition
```

rubocop-rspec

```
gem 'rubocop-rspec'
```

timecop

```
gem 'timecop'
```

```
context 'after 30 days' do
  before { Timecop.freeze(Date.today + 30) }
  after { Timecop.return }

  it 'returns true' do
    expect(subject.time_dependent_method).to eq(true)
  end
end
```

Thanks!