Ruby On Rails course

Testing

Why do we test?

What is testing?

What to test?

How to test?

Unit tests

Testing small parts of your app in isolation

Unit tests

- models
- mailers
- routes
- services
- helper

Functional tests

Parts of app that cannot live in isolation

Functional tests

- controllers
- views
- mailers

Integration tests

Important workflows within your application

Integration tests

- high-level tests
- ensuring that each of the components work together
- request specs
- feature specs (using Capybara)

Testing Frameworks

Testing Frameworks

- RSpec
- Test::Unit
- minitest
- Capybara

Setup RSpec

Installation

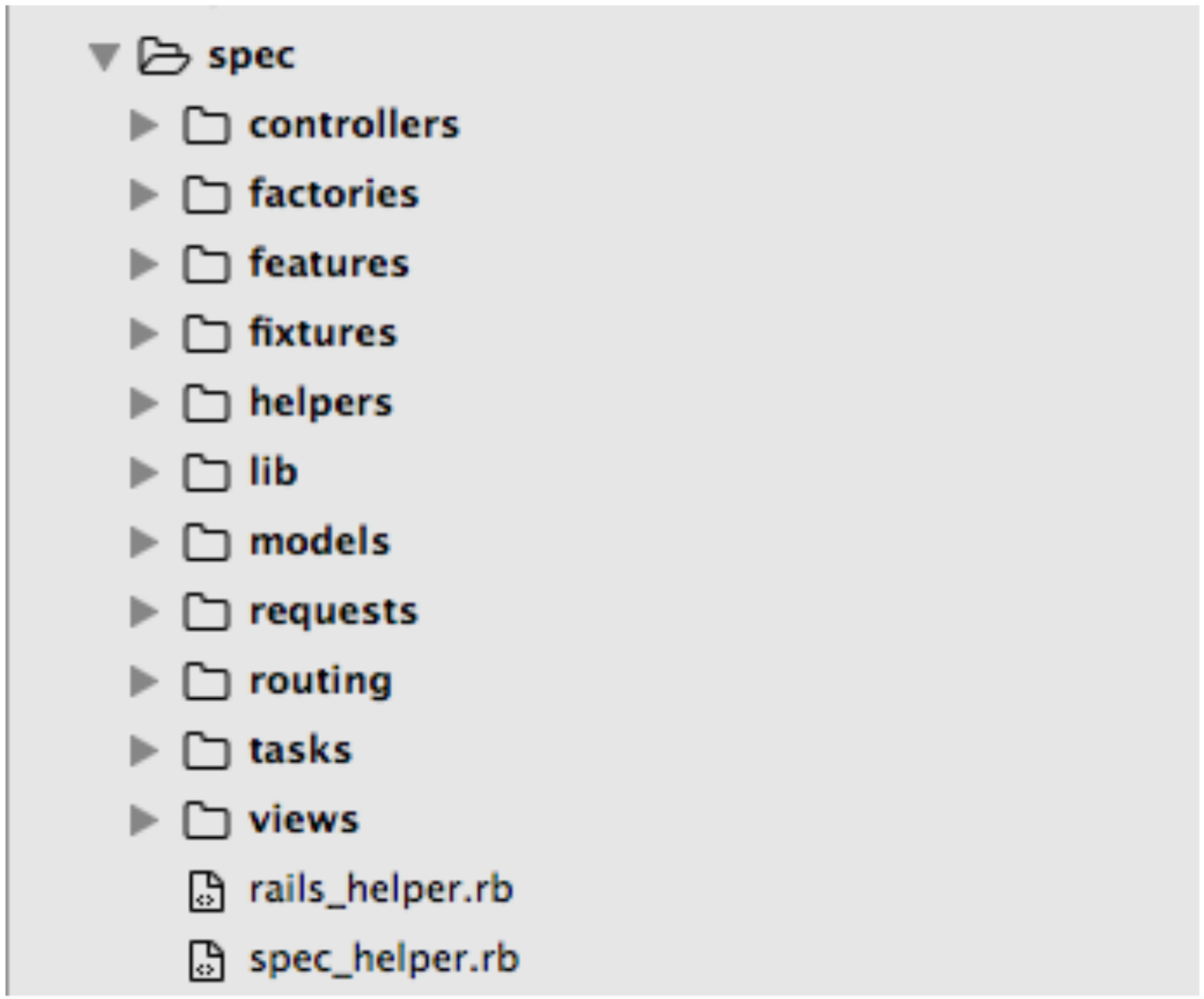
```
group :development, :test do
  gem 'rspec-rails', '~> 3.0'
end

# bundle install

# rails generate rspec:install

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

Folder structure



Test database

- database.yml
- rake db:test:prepare

```
default: &default
  adapter: sqlite3
  pool: 5
  timeout: 5000

test:
  <<: *default
  database: db/test.sqlite3
```

Run tests

```
rspec
# runs all test files

rspec spec/models
# runs all test file in this folder

rspec spec/models/subreddit_spec.rb
# runs only this test file

rspec spec/models/subreddit_spec.rb:20
# runs only the test on that line
```

Sample data

- fixtures
- factories

Fixtures

YAML-formatted predefined data

spec/fixtures/users.yml

```yaml
jan:
  first_name: Jan
  last_name: Varljen
  email: jan@productive.io
  nickname: jan.varljen
damir:
  first_name: Damir
  last_name: Svrtan
  email: damir.svrtan@infinum.hr
  nickname: DamirSvrtan
```

```
jan = users(:jan)
jan.email #jan@productive.io
```

Pros and cons

Pros

- fast
- simple
- readable

Cons

- associations
- variations
- context changes

Factories

Generate test data on the fly

spec/factories/user.rb

```ruby
FactoryGirl.define do
  factory :user do
    first_name 'first name'
    last_name 'last name'
    sequence(:email) { |n| "user#{n}@email.com" }
    nickname 'nickname'
  end
end

user = FactoryGirl.create(:user, email: 'jan@productive.io')
user.email #jan@productive.io
```

Defining associations

```ruby
FactoryGirl.define do
  factory :post do
    title 'title of the post'
    user
  end
end
```

```ruby
post = FactoryGirl.create(:post)
post.user.email #user1@email.com

user = FactoryGirl.create(:user, email: 'jan@productive.io')
post = FactoryGirl.create(:post, user: user)
post.user.email #jan@productive.io
```

Build vs Create

```
post = FactoryGirl.create(:post)
post.persisted? #true
```

```
post = FactoryGirl.build(:post)
post.persisted? #false
```

Installation

```
group :development, :test do
  gem 'factory_girl_rails'
end
```

```
# bundle install
```

Let’s write some tests!

Model test

What to test?

- validations
- scopes
- methods
- callbacks

app/models/user.rb

```ruby
class User < ActiveRecord::Base
  validates :first_name, presence: true
  validates :last_name, presence: true
  validates :email, presence: true

  scope :alphabetical, -> { order(first_name: :asc, last_name: :asc) }

  def full_name
    "#{first_name} #{last_name}"
  end
end
```

spec/factories/user.rb

```
FactoryGirl.define do
  factory :user do
    first_name "Jan"
    last_name "Varljen"
    email "jan@productive.io"
  end
end
```

spec/models/user_spec.rb

```ruby
require 'rails_helper'

RSpec.describe User, type: :model do
  pending "add some examples to (or delete) #{__FILE__}"
end
```

spec/models/user_spec.rb

```ruby
RSpec.describe User, type: :model do
  let(:user) { FactoryGirl.build(:user) }

  describe '#full_name' do
    it 'concatenates first and last name' do
      expect(user.full_name).to eq('Jan Varljen')
    end
  end
end
```

spec/models/user_spec.rb

```ruby
describe '.alphabetical' do
  let(:gabrijel) {
    FactoryGirl.create(:user, first_name: 'Gabrijel', last_name: 'Skoro')
  }
  let(:damir) {
    FactoryGirl.create(:user, first_name: 'Damir', last_name: 'Svrtan')
  }
  let(:stjepan) {
    FactoryGirl.create(:user, first_name: 'Stjepan', last_name: 'Hadjic')
  }
  it 'orders alphabetically' do
    expect(User.alphabetical).to eq([damir, gabrijel, stjepan])
  end
end
```

spec/models/user_spec.rb

```ruby
RSpec.describe User, type: :model do
  let(:user) { FactoryGirl.build(:user) }

  context 'when missing first_name' do
    let(:user) { FactoryGirl.build(:user, first_name: nil) }
    it 'validates presence of first_name' do
      expect(user.valid?).to eq(false)
    end
  end
end
```

spec/models/user_spec.rb

```ruby
RSpec.describe User, type: :model do
  let(:user) { FactoryGirl.build(:user) }

  context 'when missing first_name' do
    let(:user) { FactoryGirl.build(:user, first_name: nil) }
    it 'validates presence of first_name' do
      expect(user.valid?).to eq(false)
      expect(user.errors[:first_name]).to eq(["can't be blank"])
    end
  end
end
```

spec/models/user_spec.rb

```ruby
RSpec.describe User, type: :model do
  let(:user) { FactoryGirl.build(:user) }

  context 'when missing first_name' do
    before { user.first_name = nil }
    # after {}
    it 'validates presence of first_name' do
      expect(user.valid?).to eq(false)
      expect(user.errors[:first_name]).not_to eq(nil)
    end
  end
end
```

Controller test

What to test?

- response
  - status
  - renders
  - redirects
- creating/deleting resources

app/controllers/users_controller.rb

```ruby
class UsersController < ApplicationController
  def new
    @user = User.new
  end

  def create
    @user = User.new(user_params)

    if @user.save
      redirect_to @user
    else
      render :new
    end
  end

  private

  def user_params
    params.require(:user).permit(:first_name, :last_name, :email)
  end
end
```

spec/controllers/users_controller_spec.rb

```ruby
require 'rails_helper'

RSpec.describe UsersController, type: :controller do
  pending "add some examples to (or delete) #{__FILE__}"
end
```

spec/controllers/users_controller_spec.rb

```ruby
RSpec.describe UsersController, type: :controller do
  let(:user) { FactoryGirl.build(:user) }
  let(:valid_attributes) {
    { first_name: user.first_name, last_name: user.last_name, email: user.email }
  }
  let(:invalid_attributes) {
    { first_name: '', last_name: '', email: '' }
  }
end
```

spec/controllers/users_controller_spec.rb

```ruby
RSpec.describe UsersController, type: :controller do

  describe "GET #new" do
    it "renders new" do
      get :new
      expect(response).to be_success
      expect(response).to render_template(:new)
    end
  end
end
```

spec/controllers/users_controller_spec.rb

```ruby
describe "POST #create" do
  context "when valid" do
    it "creates a new User" do
      expect {
        post :create, user: valid_attributes
      }.to change(User, :count).by(1)
    end
    it "redirects to the created user" do
      post :create, user: valid_attributes
      expect(response).to redirect_to(User.last)
    end
  end
  context "with invalid" do
    it "re-renders new" do
      post :create, user: invalid_attributes
      expect(response).to render_template(:new)
    end
  end
end
```

Shoulda matchers

One-liners for testing common Rails functionality

Shoulda matchers

```
describe Post do
  it { should belong_to(:user) }
  it { should have_many(:comments).dependent(:destroy) }
  it { should validate_presence_of(:title) }
end
```

Code coverage

My code coverage is better than yours

Code coverage

- simplecov gem (https://github.com/colszowka/simplecov)

TDD

Test Driven Development

TDD

- test first approach
- Red - Green - Refactor

Resources

- https://github.com/howaboutwe/rspec-style-guide
- https://relishapp.com/rspec/rspec-rails/docs
- https://github.com/thoughtbot/shoulda-matchers
- https://github.com/thoughtbot/factory_girl

Announcement

31.07.2015. u 17:00 dodjela diploma

29.07.2015. do 08:00 hard deadline za predaju svih zadaća