Лекция 15
Интеграционное тестирование
Кастомные действия в AA

Типы тестов в Rails

- Unit Tests (Тесты моделей)
- Функциональные тесты (Тесты контроллеров)
- Интеграционные тесты

Fixtures

```
#/test/fixtures/questions.yml
simple_question:
  title: I have a question!
  body: Please, help
```

```
rails generate test_unit:model
question title:string body:text
```

FactoryBot

```ruby
FactoryBot.define do
  factory :user do
    id {1}
    email {"test@user.com"}
    password {"qwerty"}
  end
end
```

Генерация тестовых данных

```ruby
FactoryBot.define do
  factory :category do
    name { Faker::Lorem.word.capitalize }
  end
end
```

```ruby
FactoryBot.define do
  factory :category do
    sequence(:name) { |n| "Category number #{n}" }
  end
end
```

Подготовка к тесту

```ruby
# called before every single test
setup do
  @article = articles(:one)
end

# called after every single test
teardown do
  Rails.cache.clear
end
```

```ruby
class UserFlowsTest < ActionDispatch::IntegrationTest
  test "login and browse site" do
    # login via https
    https!
    get "/login"
    assert_response :success

    post_via_redirect "/login", username: users(:david).username,
                                password: users(:david).password
    assert_equal '/welcome', path
    assert_equal 'Welcome david!', flash[:notice]

    https!(false)
    get "/articles/all"
    assert_response :success
    assert assigns(:articles)
  end
end
```

Проблема

- Работа отдельных модулей не гарантирует работу системы
- Система склонна деградировать
- Ручное тестирование занимает много времени

Решение

Нужно автоматизировать процесс:

1. Открыть браузер
2. Нажать на кнопки, ввести данные
3. Убедиться, что нас странице именно то, что ожидалось

Selenium Web Driver

```
require "selenium-webdriver"

driver = Selenium::WebDriver.for :firefox
driver.navigate.to "http://google.com"

element = driver.find_element(name: 'q')
element.send_keys "Hello WebDriver!"
element.submit

puts driver.title

driver.quit
```

Възможности

```ruby
# execute arbitrary javascript
puts driver.execute_script("return window.location.pathname")

# pass elements between Ruby and JavaScript
element = driver.execute_script("return document.body")
driver.execute_script("return arguments[0].tagName", element)
# wait for a specific element to show up
wait = Selenium::WebDriver::Wait.new(timeout: 10) # seconds
wait.until { driver.find_element(id: "foo") }
```

Возможности

```
# switch to a frame
driver.switch_to.frame "some-frame" # id
driver.switch_to.frame driver.find_element(id:
                                'some-frame') # frame element

# repositioning and resizing browser window:
driver.manage.window.move_to(300, 400)
driver.manage.window.resize_to(500, 800)
driver.manage.window.maximize
```

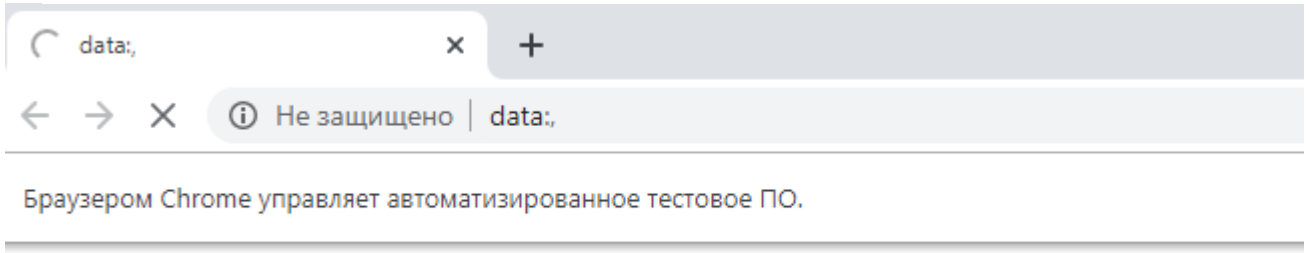
Настраиваем Rails

```
require "test_helper"

class ApplicationSystemTestCase < ActionDispatch::SystemTestCase
  driven_by :selenium,
      using: :chrome,
      screen_size: [1400, 1400]
end
```

Системные тесты

```ruby
class QuestionsTest < ApplicationSystemTestCase
  test "getting auth" do
    visit home_path
    assert_selector ".btn-primary",
                    text: (I18n.t :authorize)
  end
end
require 'test_helper'
```




```
rails test:system
```

Важность семантической верстки

```
.btn.btn-primary.align-self-auto id='enter'
  =t :authorize
```

```
click_on (I18n.t :authorize)
```

```
Minitest::UnexpectedError: Capybara::ElementNotFound: Unable to find link or button "Войти"
    test/system/questions_test.rb:7:in `block in <class:QuestionsTest>'
test/system/questions_test.rb:7:in `block in <class:QuestionsTest>'
```

```
find('div.btn-primary').click
```

Заглушки для внешних запросов

```ruby
require 'webmock/rspec'
WebMock.disable_net_connect!(allow_localhost: true)
```

```ruby
# spec/spec_helper.rb
RSpec.configure do |config|
  config.before(:each) do
    stub_request(:get, /api.github.com/).
        with(headers: {'Accept'=>'*/*', 'User-Agent'=>'Ruby'}).
        to_return(status: 200, body: "stubbed response", headers: {})
  end
end
```

Gherkin

```
Feature: Guess the word

  # The first example has two steps
  Scenario: Maker starts a game
    When the Maker starts a game
    Then the Maker waits for a Breaker to join

  # The second example has three steps
  Scenario: Breaker joins a game
    Given the Maker has started a game with the word "silky"
    When the Breaker joins the Maker's game
    Then the Breaker must guess a word with 5 characters
```

Пишем шаги

```
Given(/^I am on the homepage$/) do
  visit root_path
end

When(/^I click the provided link$/) do
  click_on "js-click-me"
end

Then(/^I should see the link click confirmation$/) do
  expect(page).to have_content("Link Clicked")
end
```

Тестирование скриншотами и снэпшотами

- Исключить длинные описания страниц
- Получить начальные данные автоматические
- Желательно сравнивать не картинки

Скриншоты в Rails

`take_screenshot`

`take_failed_screenshot` – на этапе `before_teardown`

Почему вообще Rails занимается клиентским тестированием?

- Jest
- Jasmine
- Puppeteer

Тестирование с Jest

```
yarn add --dev jest
```

```
const sum = require('./sum');

test('adds 1 + 2 to equal 3', () => {
    expect(sum(1, 2)).toBe(3);
});
```

Тестирование с Puppeteer

```
npm i puppeteer
```

```
const puppeteer = require('puppeteer');

(async () => {
    const browser = await puppeteer.launch();
    const page = await browser.newPage();
    await page.goto('https://example.com');
    await page.screenshot({path: 'example.png'});

    await browser.close();
})();
```

Jest + Puppeteer

```
yarn add --dev jest-puppeteer
```

```
describe('Google', () => {
    beforeAll(async () => {
        await page.goto('https://google.com');
    });

    it('should be titled "Google"', async () => {
        await expect(page.title()).resolves.toMatch('Google');
    });
});
```

Кастомные действия в AA

```ruby
ActiveAdmin.register Post do

  collection_action :import_csv, method: :post do
    # Do some CSV importing work here...
    redirect_to collection_path, notice: "CSV imported successfully!"
  end

end
```

post /admin/posts/import_csv

Кастомное действие с одним объектом

```
ActiveAdmin.register User do

  member_action :lock, method: :put do
    resource.lock!
    redirect_to resource_path, notice: "Locked!"
  end

end
```

Добавляем кнопки

```ruby
action_item :view, only: :show do
  link_to 'View on site', post_path(post) if post.published?
end


action_item :super_action,
            only: :show,
            if: proc{ current_admin_user.super_admin? } do
  "Only display this to super admins on the show screen"
end
```

Модификации контроллера

```ruby
ActiveAdmin.register Post do

  controller do
    # This code is evaluated within the controller class

    def define_a_method
      # Instance method
    end
  end

end
```

Страница без модели

```ruby
# app/admin/calendar.rb
ActiveAdmin.register_page "Calendar" do
  content do
    render partial: 'calendar'
  end
end

# app/views/admin/calendar/_calendar.html.arb
table do
  thead do
    tr do
      %w[Sunday Monday Tuesday Wednesday Thursday Friday Saturday].each &method(:th)
    end
  end
  tbody do
    # ...
  end
end
```

Кастомные пути

```ruby
# Available at /today/calendar
ActiveAdmin.register_page "Calendar", namespace: :today

# Available at /calendar
ActiveAdmin.register_page "Calendar", namespace: false
```

A11y

- Поддержка пользователей с ограниченными возможностями
- https://wave.webaim.org/

Замещающий текст для изображений

Изображения функциональные

```
<%= image_tag “company-logo”, alt: “Company Name” %>
```

Изображения декоративные

```
<%= image_tag “background”, alt: “” %>
```

Заголовки и формы

- Заголовки должны быть уникальными
- Все поля форм должны быть внутри тега form
- Для каждого поля должен быть элемент label

Ссылки

- https://habr.com/ru/post/423123/
- https://guides.rubyonrails.org/testing.html
- https://activeadmin.info/8-custom-actions.html