Testing on Rails

Introducing RSpec

RSpec is the original Behaviour Driven Development framework for Ruby

RSpec on Rails

- Gemfile

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

group :test do
  gem 'capybara'
  gem 'selenium-webdriver'
  gem 'factory_girl_rails'
  gem 'cucumber-rails',  :require => false
  gem 'database_cleaner',
     github: 'bmabey/database_cleaner'
end
```

  - bundle install
  - rails generate rspec:install

RSpec Testing

Unit Testing	Rspec Model Test
Unit Testing	Rspec Controller Test
Integration Testing	WebPage Test (Rspec, Capybara, Selenium)

RSpec Syntax

* file: spec user_spec.rb
* statement: describer - it - should

```
describe AA do
   it 'should do something' do
      something.should ...
   end
end
```

Model Testing on RSpec

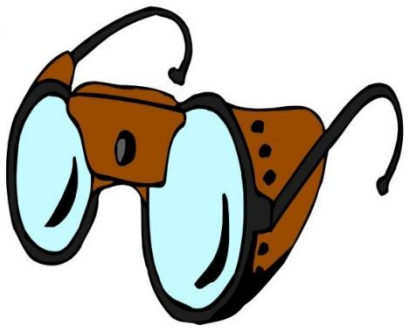
- file: spec/models/post_spec.rb

```
describe "testing model:post" do
  it "should respond to 'name' and 'title" do
    @post = Post.new(name: "Mike", title:"my first post")
    expect(@post).to respond_to(:name)
    expect(@post).to respond_to(:title)
  end
  it "name is empty" do
    @post = Post.new(name: "", title:"my first post")
    expect(@post).not_to be_valid
  end
  it "title is too short" do
    @post = Post.new(name: "Mike", title:"1")
    expect(@post).not_to be_valid
  end
end
```

bundle exec rspec spec/

Red-Yellow-Green Analysis

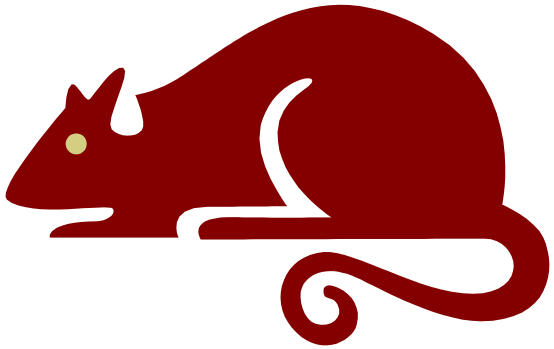
- Rspec colors steps
  - Green for passing
  - Yellow for not yet implemented
  - Red for failing
  - Goal: Make all steps green for pass



Web Page Testing on RSpec

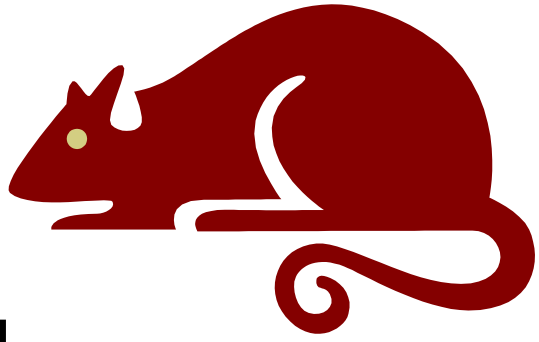
- Capybara
  - Gemfile
    - Gem ‘capybara’
  - spec/rails_helper.rb

```
RSpec.configure do |config|
  .
  .
  .
  config.include Capybara::DSL
end
```



Fake User to Test?

- Need tool that pretends to be the user to follow scenarios of user stories
- **Capybara simulates browser**
  - Can interact with app to receive pages
  - Parse the HTML
  - Submit forms as a user would



Web Page Testing on RSpec

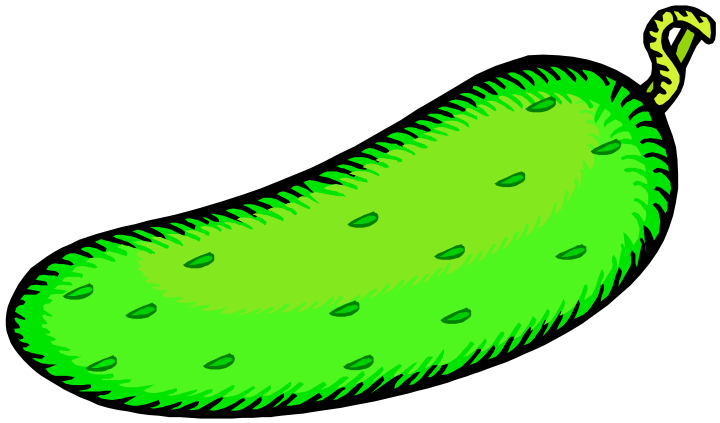
- rails generate integration_test static_pages
- file: spec/requests/static_pages_spec.rb

```
#----------Testing Home Page------------------------------
describe "Home page" do
   it "should have the content 'Hello'(not 'Home')
        and a link 'My Blog'" do
      visit root_path
      expect(page).to have_content('Hello')
      expect(page).not_to have_content('Home')
      expect(page).to have_link('My Blog', href: posts_path)
   end
end
```

Web Page Testing on RSpec

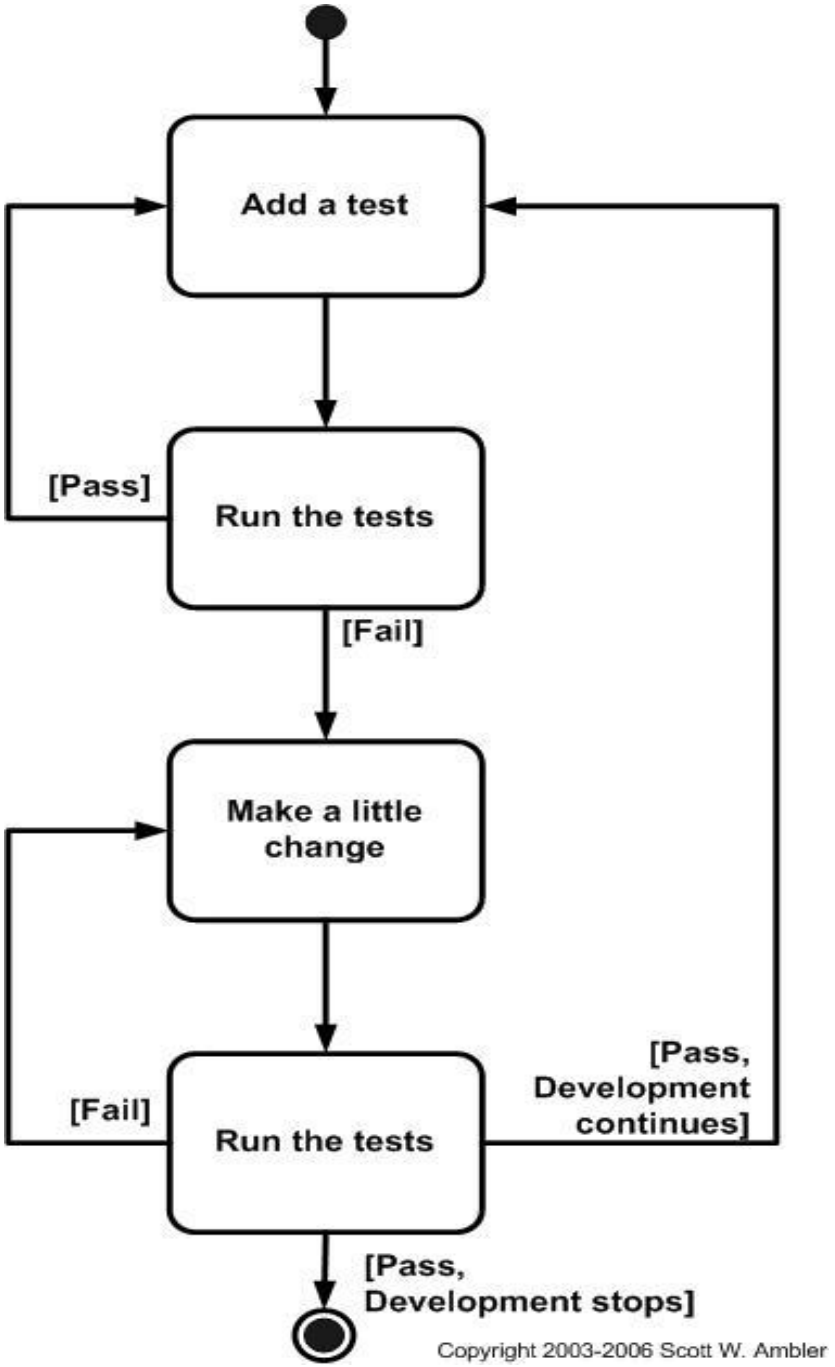
```
#----------Subject----------------------------------------
subject { page }

#----------Testing Posts Page-----------------------------
describe "Posts page" do
  before { visit posts_path }
  it { should have_content('Posts') }
  it { should have_link('New Post', href: new_post_path) }
  describe "should be able to create a post" do
    before {click_link('New Post')}
    it { should have_content('New Post') }
  end
end
```

Introducing Cucumber



TDD

Test-Driven Development



BDD

- ***Behaviour-Driven Development***
  - *Behaviour-Driven Development is a second-generation, multiple-stakeholder, multiple-scale, high-automation, agile methodology. It describes a cycle of interactions with well-defined outputs, resulting in the delivery of working, tested software that matters.*

    --November 2009 in London, **Dan North**

Example User Story

```
Feature: Signing in
```

1 Feature

≥1 Scenarios / Feature

```
Scenario: Unsuccessful signin
```

```
Given a user visits the signin page
When he submits invalid signin information
Then he should see an error message
```

Steps / Scenario

Example User Story (cont'd)

```
Scenario: Successful signin
  Given a user visits the signin page
  And the user has an account
  When the user submits valid signin information
  Then he should see his profile page
  And he should see a signout link
```

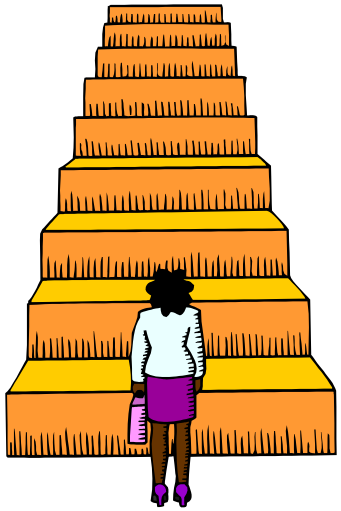
User Story, Feature, and Steps

- User story: refers to single feature
- Feature: ≥1 scenarios that show different ways a feature is used
  - Keywords Feature and Scenario identify respective components
  - Kept in .feature files

- Scenario: 3 - 8 steps that describe scenario
- Step definitions: Ruby code to test steps
  - Kept in X_controller.rb files

5 Step Keywords

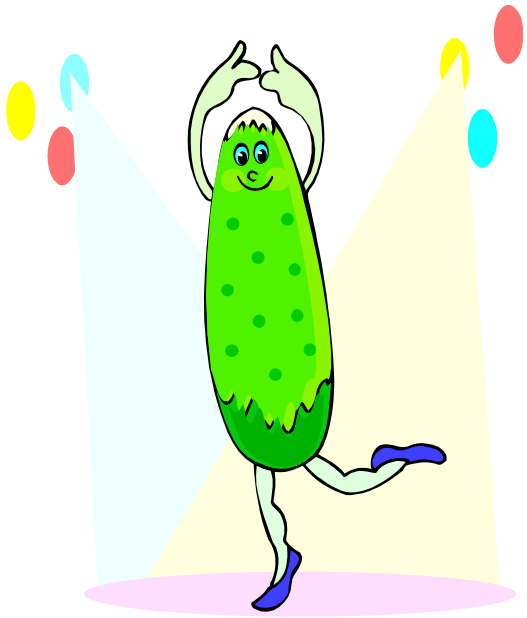
1. Given steps represent state of world before event: *preconditions*
2. When steps represent event
   - e.g., simulate user pushing a button
3. Then steps represent expected *postconditions*; check if true
4. / 5. And & But extend previous step



Testing with cucumber

- Run the test
  - bundle exec cucumber features
  - Pending

```
2 scenarios (2 undefined)
8 steps (8 undefined)
0m0.384s
```



Write steps for the feature

features/step_definitions/authentication_steps.rb

- For scenario 1:

```
Given /^a user visits the signin page$/ do
  visit signin_path
end

When /^he submits invalid signin information$/ do
  click_button "Sign in"
end

Then /^he should see an error message$/ do
  expect(page).to have_selector('div.alert.alert-error')
end
```

Write steps for the feature (cont'd)

For scenario 2:

```
Given /^the user has an account$/ do
  @user = User.create(name: "Example User", email: "user@example.com",
                      password: "foobar", password_confirmation: "foobar")
end

When /^the user submits valid signin information$/ do
  fill_in "Email",    with: @user.email
  fill_in "Password", with: @user.password
  click_button "Sign in"
end

Then /^he should see his profile page$/ do
  expect(page).to have_title(@user.name)
end
```

Thank You