Testing With Selenium

Jake Sorce / Dave Jungst - Devpoint Labs

What is Selenium?

Selenium automates browsers. That's it! What you do with that power is entirely up to you.

Why do I want to use Selenium?

- Testing your models and controllers are great because you are testing the backend logic of your app, but what about the frontend user experience?
- Selenium automates a user clicking around in your application
- Catch view bugs, dead buttons, wrong routes, javascript errors, etc..

What do I need to do to get started?

- Let's start simple with the Firefox IDE:
  - Open Firefox
  - Http://docs.seleniumhq.org/download/
  - Cmd+f to search for the keyword Firefox
  - Should take you about ⅓ of the page down
  - Click on the version number to download the Selenium plugin for Firefox
  - Restart Firefox

How do I use the Firefox IDE?

- Open Firefox
- Go to the website you want to automate
- Click on the Selenium IDE button at the top or go to plugins and launch the IDE from there
- Click on the record button at the top right
- Start using the website for a specific test like a normal user would. (eg. login / logout test)
- If you are running tests against your local install of your website, start your server in test mode so you don't disturb your development data. (rails s -p 3001 -e test)

Thats awesome, lets do it in Rails with RSpec!

- Why do we want to do this in Rails?
  - clear out the database
  - write more complex tests
  - write our own tests
  - cleanup the test that the IDE generated
  - store our tests in our version control system so our other team members can run them

Rails Setup

```
rails new selenium_example -T -d postgresql

group :development, :test do
        gem 'rspec-rails'
        gem 'capybara'
        gem 'selenium-webdriver'
        gem 'database_cleaner'
end

bundle install

rails generate rspec:install
```

rails_helper.rb Setup

- add these 2 lines at the top of your file where the other requires are
  - require 'rspec/rails'
  - require 'capybara/rspec'

rails_helper.rb setup cont...

```
RSpec.configure do |config|
 # Remove this line if you're not using ActiveRecord or ActiveRecord fixtures
 config.fixture_path = "#{::Rails.root}/spec/fixtures"
 config.use_transactional_fixtures = false

 config.before(:each) do
  DatabaseCleaner.strategy = :transaction
 end

 config.before(:each, :js => true) do
  DatabaseCleaner.strategy = :truncation
 end

 config.before(:each) do
  DatabaseCleaner.start
 end

 config.after(:each) do
  DatabaseCleaner.clean
 end
config.infer_spec_type_from_file_location!
end
```

Rails Setup Cont...

- add a couple of models
- add controllers for those models and fill out all of the CRUD actions
- code out the views
- start writing view specs

Spec Folder Setup

- Create a folder inside of the spec folder and name it features

- Create your spec file inside of the features folder. Make sure it ends with _spec.rb

Basic Spec Setup

```
require 'rails_helper'

feature 'Empty Classrooms', :js => true do
  before(:each) do
    visit root_path
  end

  scenario 'correct empty message' do
    expect(page).to have_content('No Classrooms Found. Please Create One!')
  end

  scenario 'create classrooms link exists' do
    expect(page).to have_content('New Classroom')
  end
end
```

Capybara DSL

- https://github.com/jnicklas/capybara#the-dsl

- Navigating
- Clicking links and buttons
- Interacting with forms
- Querying
- Finding
- Scoping
- Working with windows
- Scripting
- Modals
- Debugging
- Gotchas

Navigating

visit('/projects')

=> url version

visit(post_comments_path(post))

=> rails helper version

Navigation Expectations

expect(current_path).to eq(post_comments_path(post))

Clicking links and buttons

click_link('id-of-link')

click_link('Link Text')

click_button('Save')

click_on('Link Text') # clicks on either links or buttons

click_on('Button Value')

Interacting with forms

```
fill_in('First Name', :with => 'John')

fill_in('Password', :with => 'Seekrit')

fill_in('Description', :with => 'Really Long Text...')

choose('A Radio Button')

check('A Checkbox')

uncheck('A Checkbox')

attach_file('Image', '/path/to/image.jpg')

select('Option', :from => 'Select Box')
```

Querying

page.has_selector?('table tr')

page.has_css?('table tr.foo')

page.has_content?('foo')

Querying Expectations

expect(page).to have_selector('table tr')

expect(page).to have_css('table tr.foo')

expect(page).to have_content('foo')

Finding

find_field('First Name').value

find_link('Hello', :visible => :all).visible?

find_button('Send').click

find("#overlay").find("h1").click

all('a').each { |a| a[:href] }

Finding Expectations

```
find('#navigation').click_link('Home')

expect(find('#navigation')).to have_button('Sign out')
```

Scoping

```
within("li#employee") do
  fill_in 'Name', :with => 'Jimmy'
end

within_fieldset('Employee') do
  fill_in 'Name', :with => 'Jimmy'
end

within_table('Employee') do
  fill_in 'Name', :with => 'Jimmy'
end
```

Working with windows

```
facebook_window = window_opened_by do
  click_button 'Like'
end

within_window facebook_window do
  find('#login_email').set('a@example.com')
  find('#login_password').set('qwerty')
  click_button 'Submit'
end
```

Scripting

```
page.execute_script("$('body').empty()")
```

```
result = page.evaluate_script('4 + 4');
```

Modals

```
accept_alert do
  click_link('Show Alert')
end

dismiss_confirm do
  click_link('Show Confirm')
end

accept_prompt(with: 'Linus Torvalds') do
  click_link('Show Prompt About Linux')
end
```

Modals Expectations

```
message = accept_prompt(with: 'Linus Torvalds') do
  click_link('Show Prompt About Linux')
end


expect(message).to eq('Who is the chief architect of Linux?')
```

Debugging

save_and_open_page

print page.html

page.save_screenshot('screenshot.png')

save_and_open_screenshot

Gotchas

- Access to session and request is not possible from the test, Access to response is limited.

- Access to Rails specific stuff (such as controller) is unavailable, since we're not using Rails' integration testing.

Run your specs!

- bundle exec rspec spec/features
  - this command should open up a new browser and start running your view specs

Project

- Create a new rails application with Capybara and RSpec installed
- Add a Scientist model, add a Robot model
- A Scientist has_many :robots
- A Robot belongs_to :scientist
- Fill out the CRUD actions for each
- Use Capybara, Selenium with RSpec to test your views

Resources

- https://github.com/jnicklas/capybara#the-dsl
- http://www.rubydoc.info/github/jnicklas/capybara/master
- http://humblecoder.me/tag/capybara-tutorial/