RSPEC Part-2

Hiep Le Tuan veu, Sun-Asterisk
le.tuan.hiep@sun-asterisk.com

Ngày 29 tháng 2 năm 2020

Mục lục

I. Controller specs

Controller specs được đánh dấu bởi *:type => :controller* hoặc thiết lập *config.infer_spec_type_from_file_location!* trong *spec/controller*
Một controller spec là một wrapper của Rails functional test. Nó cho phép ta mô phỏng một http request trong mỗi example và kỳ vọng một response trả về cụ thể. Ví dụ:

- render templates
- redirects
- Biến đối tượng được gán trong controller được shared với view
- cookies gửi lại cùng với response

Để chỉ định kết quả trả về ta có thể sử dụng:

- standard rspec matchers **(expect(response.status).to eq(200))**
- standard test/unit assertions **(assert_equal 200, response.status)**
- rails assertions **(assert_response 200)**
- rails-specific matchers:
  - render_template

```
expect(response).to render_template(:new)
```

  - redirect_to

```
expect(response).to redirect_to(location)
```

  - have_http_status

```
expect(response).to have_http_status(:created)
```

  - be_a_new

```
    expect(assigns(:widget)).to be_a_new(Widget)
```

Ví dụ:

```
RSpec.describe TeamsController do
  describe "GET index" do
    it "assigns @teams" do
      team = Team.create
      get :index
      expect(assigns(:teams)).to eq([team])
    end

    it "renders the index template" do
      get :index
      expect(response).to render_template("index")
    end
  end
end
```

1. Cookies

Controller specs cung cấp 1 số cách truy nhập cookies:

```
@request.cookies['key']
@response.cookies['key']
cookies['key']
```

Chú ý: Với railss-3.0.x và 3.1 có cách xử lý khác một chút, vì vậy để tránh nhầm lẫn, ta sử dụng hướng dẫn sau:

- Truy nhập cookies thông qua đối tượng request và response trong spec.
  - Sử dụng request.cookies trước hành động để thiết lập trạng thái
  - Sử dụng response.cookies sau khi hành động để chỉ định đầu ra cụ thể
- Sử dụng đối tượng cookies trong controller action.
- Sử dụng String keys.

```
# spec
request.cookies['foo'] = 'bar'
get :some_action
expect(response.cookies['foo']).to eq('modified bar')

# controller
def some_action
  cookies['foo'] = "modified #{cookies['foo']}"
end
```

Tại sao sử dụng Strings thay Symbols?
Đối tượng cookies trong spec được cung cấp bởi Rack, và không hô trợ truy cập thông minh (ví dụ: :foo và “foo" là những keys khác nhau).
Thay đổi này diễn ra trong rails 3.1, vì vậy ta có thể sử dụng symbol keys nhưng khuyến nghị nên dùng string key để đảm bảo tính nhất quán.
Tại sao không sử dụng cookies method? Cookies method kết hợp cả cookies request và cookies respond. Nó có thể gây ra sự nhầm lẫn khi thiết lập cookies trong example để thiết lạp trạng thái cho controller action.

```
# does not work in rails 3.0.0 > 3.1.0
  cookies['foo'] = 'bar' # this is not visible in the controller
```

```
get :some_action
```

2. Controller spec

- 1 Method pass example:

```
#spec/controllers/widgets_controller_spec.rb
require "rails_helper"

RSpec.describe WidgetsController, :type => :controller do
   describe "GET index" do
        it "has a 200 status code" do
        get :index
        expect(response.status).to eq(200)
   end
end
end
```

- Controller được thiết lập toàn cục trước khi hooks

```
#spec/controllers/widgets_controller_spec.rb
require "rails_helper"

RSpec.configure {|c| c.before { expect(controller).not_to be_nil }}

RSpec.describe WidgetsController, :type => :controller do
   describe "GET index" do
      it "doesn't matter" do
      end
   end
end
```

- Thiết lập một content type khác cho example json (request type) @rails_pre_5

```
#spec/controllers/widgets_controller_spec.rb
require "rails_helper"

RSpec.describe WidgetsController, :type => :controller do
   describe "responds to" do
      it "responds to html by default" do
           post :create, { :widget => { :name => "Any Name" } }
           expect(response.content_type).to eq "text/html"
      end

      it "responds to custom formats when provided in the params" do
           post :create, { :widget => { :name => "Any Name" }, :format => :json }
           expect(response.content_type).to eq "application/json"
      end
   end
end
```

- Thiết lập một content type khác cho example json (request type) @rails_post_5

```
#spec/controllers/widgets_controller_spec.rb
require "rails_helper"

RSpec.describe WidgetsController, :type => :controller do
   describe "responds to" do
```

```
    it "responds to html by default" do
        post :create, :params => { :widget => { :name => "Any Name" } }
        expect(response.content_type).to eq "text/html"
    end

    it "responds to custom formats when provided in the params" do
        post :create, :params => { :widget => { :name => "Any Name" }, :format => :json }
        expect(response.content_type).to eq "application/json"
    end
  end
end
```

3. Mặc định views là các stubs

Mặc đinh controller specs stub views với 1 teamplate sẽ hiển thị string thay cho views ứng dụng. Điều này cho phép ta chỉ định template action nào nên được render ra.

- Kỳ vọng template được hiển thị bởi controller action

```
require "rails_helper"

RSpec.describe WidgetsController, :type => :controller do
  describe "index" do
    it "renders the index template" do
        get :index
        expect(response).to render_template("index")
        expect(response.body).to eq ""
    end
    it "renders the widgets/index template" do
        get :index
        expect(response).to render_template("widgets/index")
        expect(response.body).to eq ""
    end
  end
end
```

- Kỳ vọng temple rỗng khi view path thay đổi khi runtime

```
require "rails_helper"

RSpec.describe ThingsController, :type => :controller do
  describe "custom_action" do
    it "renders an empty custom_action template" do
        controller.prepend_view_path 'app/views'
      controller.append_view_path 'app/views'
      get :custom_action
        expect(response).to render_template("custom_action")
        expect(response.body).to eq ""
    end
  end
end
```

- Kỳ vọng một temple thực với render_views khi view path thay đổi tại runtime

```
require "rails_helper"

RSpec.describe ThingsController, :type => :controller do
  render_views
```

```
    it "renders the real custom_action template" do
      controller.prepend_view_path 'app/views'
      get :custom_action
      expect(response).to render_template("custom_action")
      expect(response.body).to match(/template for a custom action/)
    end
  end
```

Tài liệu

[1] Relish Team, *https://relishapp.com/rspec*,