RSPEC Part-1

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

Ngày 15 tháng 2 năm 2020

I. Khởi tạo

Các file rspec thường được tạo cùng các thành phần ứng dụng khác. Ví dụ: *rails generate model* cũng sẽ tạo luôn file Rspec cho model.
Những file spec này không viết sẵn các trường hợp test mà chỉ đảm bảo việc file spec có cấu trúc mặc định của Rails.
Các file rspec cũng có thể được tạo ra độc lập. Ví dụ: *rails generate rspec:model user* sẽ tạo ra file spec mới trong *spec/models/user_ spec.rb*. Câu lệnh khởi tạo có sẵn với:

- scaffold
- model
- controller
- helper
- view
- mailer
- observer
- integration
- feature
- job

II. Giao dịch (Transactions)

Khi bạn chạy câu lệnh *rails generate rspec:install*, file textitspec/rails_helper.rb sẽ có phần thiết lập sau:

```
RSpec.configure do |config|
  config.use_transactional_fixtures = true
end
```

Tên của thiết lập có một chút không chính xác với chức năng nó đại diện. Thực chất, thiết lập này trong Rails mang ý nghĩa: "Chạy mọi phương thức test trong 1 **transaction**." Còn trong ngữ cảnh rspec-rails: "Chạy mọi example trong 1 **transaction**."
Ý tưởng này bắt nguồn từ việc khi chạy mỗi example với một db rỗng, tạo các dữ liệu cần thiết và bỏ nó (rollback db) khi kết thúc example.
Tắt transactions: Nếu bạn muốn tự quản lý dữ liệu hoặc dùng một tool khác như database_cleaner thay thế thì chỉ việc chỉnh thiết lập về false.

1. Data được tạo trong before(:example) sẽ bị rolled back

Tất cả dữ liệu tạo trong before(:example) sẽ bị rolled back khi kết thúc example. Điều này có nghĩa mỗi example sẽ độc lập với nhau, không bị ảnh hưởng bởi nhau. Ví dụ:

```
describe Widget do
  before(:example) do
    @widget = Widget.create
  end

  it "does something" do
    expect(@widget).to do_something
  end

  it "does something else" do
    expect(@widget).to do_something_else
  end
end
```

Biến @widget được khởi tạo lại trước mỗi example. Mỗi example có một đối tượng riêng và khi kết thúc mỗi example, dữ liệu được rolled back. Biến @widget là biến mới hoàn toàn.

2. Data được tạo trong before(:context) sẽ bị rolled back

before(:context) được gọi trước khi transaction xảy ra. Việc này giúp giảm thời gian khởi tạo dữ liệu trước mỗi example trong group được chạy. Tuy nhiên, nó cũng khiến cho nảy sinh những vấn đề. Bạn chỉ nên dùng nó nếu bạn nắm chắc. Một vài lưu ý khi sử dụng:
- Chắc chắn xóa sạch dữ liệu trong after(:context)

```
before(:context) do
  @widget = Widget.create!
end

after(:context) do
  @widget.destroy
end
```

Nếu không làm điều này, dữ liệu sẽ vẫn tồn tại và ảnh hưởng tới các examples khác.
- Reload đối tượng trong before(:example)

```
before(:context) do
  @widget = Widget.create!
end

before(:example) do
  @widget.reload
```

Mặc dù những cập nhật của db trong mỗi example được rolled back nhưng đối tượng khong thể biết về những rolled back này. Nên nó cần reload lại để đồng bộ dữ liệu.

III. Cấu trúc thư mục

Các file specs được đặt trong cấu trúc thư mục được mô tả với mục đích của chúng:
- Model specs đặt trong thư mục spec/models
- Controller specs đặt trong thư mục spec/controllers
- Request specs đặt trong thư mục spec/requests. Thư mục này cũng có thể được đặt tên là integration hoặc api.
- Feature specs đặt trong thư mục spec/features.
- View specs đặt trong thư mục spec/views.
- Helper specs đặt trong thư mục spec/helpers.

- Mailer specs đặt trong thư mục spec/mailes.
- Routing specs đặt trong thư mục spec/routing.
- Job specs đặt trong thư mục spec/jobs.
- System specs đặt trong thư mục spec/system.

Các developers thoải mái trong việc sử dụng những dạng cấu trúc thư mục khác. Để các functions hỗ trợ rspec-rails chính xác, specs cần có metadata tương ứng phù hợ :type
- Model specs: type: :model
- Controller specs: type: :controller
- Request specs: type: :request
- Feature specs: type: :feature
- View specs: type: :view
- Helper specs: type: :helper
- Mailer specs: type: :mailer
- Routing specs: type: :routing
- Job specs: type: :job
- System specs: type: :system

Ví dụ, để khai báo spec cho ThingsController nằm ở *spec/legacy/things_controller_spec.rb* thì chỉ cần khai báo metadata *type: :controller*

```
# spec/legacy/things_controller_spec.rb
RSpec.describe ThingsController, type: :controller do
  describe "GET index" do
    Examples
  end
end
```

IV. Model specs

Model specs được đánh dấu bởi type: :model. Một model spec được bao bọc bởi ActiveSupport::TestCase và bao gồm những hành vi, quyền hạn mà nó cung cấp ngoài các hành vi và kỳ cọng của RSpec. Ví dụ:

```
RSpec.describe Post, :type => :model do
  context "with 2 or more comments" do
    it "orders them in reverse chronologically" do
      post = Post.create!
      comment1 = post.comments.create!(:body => "first comment")
      comment2 = post.comments.create!(:body => "second comment")
      expect(post.reload.comments).to eq([comment2, comment1])
    end
  end
end
```

1. Transactional examples

Mặc định, rspec chạy mỗi example trong một transaction. Bạn có thể bật/tắt những transactions này bằng thiết lập thuộc tính 'use_transactional_examples'.

- Chạy transaction (mặc định)

```
# spec/models/widget_spec.rb:
require "rails_helper"

RSpec.describe Widget, :type => :model do
  it "has none to begin with" do
    expect(Widget.count).to eq 0
  end

  it "has one after adding one" do
```

```
    Widget.create
    expect(Widget.count).to eq 1
  end

  it "has none after one was created in a previous example" do
    expect(Widget.count).to eq 0
  end
end
```

Khi chạy *rspec spec/models/widget_ spec.rb* //Kết quả: *the examples should all pass*

- Chạy transaction (cụ thể)

```
# spec/models/widget_spec.rb
require "rails_helper"

RSpec.configure do |c|
  c.use_transactional_examples = true
end

RSpec.describe Widget, :type => :model do
  it "has none to begin with" do
    expect(Widget.count).to eq 0
  end

  it "has one after adding one" do
    Widget.create
    expect(Widget.count).to eq 1
  end

  it "has none after one was created in a previous example" do
    expect(Widget.count).to eq 0
  end
end
```

Khi chạy *rspec spec/models/widget_ spec.rb* //Kết quả: *the examples should all pass*

- Tắt transaction (cụ thể)

```
# spec/models/widget_spec.rb
require "rails_helper"

RSpec.configure do |c|
  c.use_transactional_examples = false
  c.order = "defined"
end

RSpec.describe Widget, :type => :model do
  it "has none to begin with" do
    expect(Widget.count).to eq 0
  end

  it "has one after adding one" do
    Widget.create
    expect(Widget.count).to eq 1
  end

  it "has one after one was created in a previous example" do
    expect(Widget.count).to eq 1
  end

  after(:all) { Widget.destroy_all }
end
```

Khi chạy *rspec spec/models/widget_ spec.rb*
Kết quả: *the examples should all pass*
- Chạy trong transactions cố định

```
# spec/models/thing_spec.rb
require "rails_helper"

RSpec.describe Thing, :type => :model do
  fixtures :things
  it "fixture method defined" do
    things(:one)
  end
end
```

```
# spec/fixtures/things.yml
one:
  name: MyString
```

Khi chạy *rspec spec/models/thing_ spec.rb* //Kết quả: *the examples should all pass*

2. Xác minh kép

Mặc định, rspec xác minh kép không hỗ trợ các hàm dynamic trong instance_dobule. rspec-rails bật phần hỗ trợ này cho cột phương thức thông qua một phần mở rộng.
Ngữ cảnh

```
# spec/models/widget_spec.rb
require "rails_helper"

RSpec.describe Widget, :type => :model do
  it "has one after adding one" do
    instance_double("Widget", :name => "my name")
  end
end
```

Khi chạy *rspec spec/models/widget_ spec.rb* //Kết quả: *the examples should all pass*

Tài liệu

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