Ruby On Rails par Rspec

Cyril Mougel

30 Décembre 2008

1 Rspec, BDD ?
2 Rails est agile avant tout
3 La couche modèle de Rails
4 La couche controller de Rails
5 Les routes Rails
6 La couche Vue de Rails
7 les mocks ?

Behaviour Driven Development

- Méthode Agile
- Extreme programming
- TDD (Test Driven Development)
- l'empire du 'should'

1 Rspec, BDD ?

2 Rails est agile avant tout

3 La couche modèle de Rails

4 La couche controller de Rails

5 Les routes Rails

6 La couche Vue de Rails

7 les mocks ?

Test : :Unit de base dans Rails

- Dossier test créé à chaque création de projet Rails
- Stats de nombre du LOC et LOC de test

```
(in /mnt/data/home_data/perso/project/typo)
+----------------------+-------+-------+---------+---------+-----+-------+
| Name                 | Lines |   LOC | Classes | Methods | M/C | LOC/M |
+----------------------+-------+-------+---------+---------+-----+-------+
| Controllers          |  2283 |  1879 |      37 |     217 |   5 |     6 |
| Helpers              |   695 |   569 |       0 |      87 |   0 |     4 |
| Models               |  3416 |  2715 |      61 |     465 |   7 |     3 |
| Libraries            |  1322 |   982 |      32 |     149 |   4 |     4 |
| APIs                 |   436 |   353 |      17 |      23 |   1 |    13 |
| Model specs          |  1754 |  1402 |       7 |      87 |  12 |    14 |
| View specs           |   104 |    81 |       0 |       8 |   0 |     8 |
| Controller specs     |  2897 |  2226 |      21 |     174 |   8 |    10 |
| Helper specs         |    51 |    38 |       0 |       0 |   0 |     0 |
+----------------------+-------+-------+---------+---------+-----+-------+
| Total                | 12958 | 10245 |     175 |    1210 |   6 |     6 |
+----------------------+-------+-------+---------+---------+-----+-------+
  Code LOC: 6498     Test LOC: 3747     Code to Test Ratio: 1:0.6
```

- Possibilité de tester chaque couche de Rails

Mais pourquoi Rspec alors ?

- Rspec == BDD Framework
- Documentation générée plus claire que les Test : :Unit

RSpec Results — 26 examples, 0 failures
Finished in 0.022214 seconds

Stack (empty)
- should be empty
- should not be full
- should add to the top when sent #push
- should complain when sent #peek
- should complain when sent #pop

Stack (with one item)
- should not be empty
- should return the top item when sent #peek
- should NOT remove the top item when sent #peek
- should return the top item when sent #pop
- should remove the top item when sent #pop
- should not be full
- should add to the top when sent #push

- Réutilisation plus simple du comportement
- Ajout des Stories (Cucumber)

Et ca s'installe comment ?

- gem install rspec && gem install rspec-rails
- gem install rspec &&
  ./script/install git ://github.com/dchelimsky/rspec-rails.git
- gem install rspec &&
  git-submodule add vendor/plugins/rspec-rails
  git ://github.com/dchelimsky/rspec-rails.git

1 Rspec, BDD ?

2 Rails est agile avant tout

3 La couche modèle de Rails

4 La couche controller de Rails

5 Les routes Rails

6 La couche Vue de Rails

7 les mocks ?

Des classes de Mapping à la Base de donnée

```
1  class User < ActiveRecord::Base
2    has_many :products
3
4    validates_presence_of :name
5    validates_uniqueness_of :name
6    validates_format_of :email,
7      :with => /\A([^@\s]+)@((?:[-a-z0-9]+\.)+[a-z]{2,})\Z/i,
8      :on => :create
9  end
10
11 class Product < ActiveRecord::Base
12   belongs_to :user
13 end
```

Doit créer les accesseurs sur les colonnes

```
1  describe User do
2
3    before(:each) do
4      @user = User.first
5    end
6
7    it 'should access to name' do
8      @user.name.should_not be_nil
9    end
10
11 end
```

Doit valider le model

```
1   describe User do
2     describe 'create' do
3       def user_valid(options)
4         User.new({:name => 'Cyril Mougel',
5                   :email => 'cyril.mougel@gmail.com'}.merge(options))
6       end
7
8       it 'should user valid' do
9         user_valid.should be_valid
10      end
11
12      it 'should presence of name' do
13        user_valid(:name => nil).should_not be_valid
14      end
15
16      it 'should uniq name' do
17        user_valid.save
18        user_valid.should_not be_valid
19      end
20
21      it 'should not valid with bad email' do
22        user_valid(:email => 'cool').should_not be_valid
23      end
24    end
25  end
```

Doit chercher des données

```
1   describe User do
2
3     it 'should find by name' do
4       User.find_by_name('Cyril Mougel').should == users(:shingara)
5     end
6
7     it 'should find by email' do
8       User.find_by_email('cyril.mougel@gmail.com').should == users(:shingara)
9     end
10
11    it 'should find by name and email' do
12      User.find_by_name_and_email('Cyril Mougel',
13                                  'cyril.mougel@gmail.com').should == users(:shing
14    end
15
16    it 'should find all' do
17      User.all.should == [users(:shingara), users(:underflow_)]
18    end
19  end
```

Doit avoir des associations

```
1   describe User do
2     before :each do
3       @user = users(:shingara)
4     end
5
6     it 'should have 2 products' do
7       @user.should have(2).products
8       # @user.products.size == 2
9     end
10
11    it 'should have book products' do
12      @user.products[0].should == products[:book]
13    end
14  end
15
16  describe Product do
17    before :each do
18      @product = products(:book)
19    end
20
21    it 'should have user shingara' do
22      @product.user.should == users(:shingara)
23    end
24  end
```

1 Rspec, BDD ?
2 Rails est agile avant tout
3 La couche modèle de Rails
4 La couche controller de Rails
5 Les routes Rails
6 La couche Vue de Rails
7 les mocks ?

Un controlleur Rails

```
1   class UsersController < ApplicationController
2
3     def index
4       @users = User.find(:all)
5       respond_to do |format|
6         format.html # index.html.erb
7         format.xml  { render :xml => @users }
8       end
9     end
10
11    def show
12      @user = User.find(params[:id])
13      respond_to do |format|
14        format.html # show.html.erb
15        format.xml  { render :xml => @user }
16      end
17    end
18
19    def create
20      @user = User.new(params[:user])
21      respond_to do |format|
22        if @user.save
23          flash[:notice] = 'User was successfully created.'
24          format.html { redirect_to(@user) }
25        else
26          format.html { render :action => "new" }
27        end
28      end
29    end
30  end
```

Doit permettre de voir la liste des utilisateurs

```
1   describe UsersController do
2
3     describe "responding to GET index" do
4       it "should expose all users as @users" do
5         get :index
6         assigns[:users].should == [users(:shingara)]
7       end
8
9       describe "with mime type of xml" do
10        it "should render all users as xml" do
11          request.env["HTTP_ACCEPT"] = "application/xml"
12          get :index
13          response.body.should == users(:shingara).to_xml
14        end
15      end
16    end
17  end
```

Doit permettre de voir un utilisateur particulier

```
1   describe UsersController do
2     describe "responding to GET show" do
3
4       it "should expose the requested user as @user" do
5         get :show, :id => users(:shingara).id
6         assigns[:user].should == users(:shingara)
7       end
8
9       describe "with mime type of xml" do
10
11        it "should render the requested user as xml" do
12          request.env["HTTP_ACCEPT"] = "application/xml"
13          get :show, :id => users(:shingara).id
14          response.body.should == users(:shingara).to_xml
15        end
16      end
17    end
18  end
```

Doit créer un utilisateur

```
describe UsersController do
  describe "responding to POST create" do

    describe "with valid params" do

      it "should create user" do
        assert_difference 'User.count' do
          post :create, :user => {:name => 'Jean-francois',
                                  :email => 'jf@rubyfrance.org'}
          response.should redirect_to(
                                  user_url(User.find_by_name('Jean-francois')))
        end
      end

    end

    describe "with invalid params" do

      it "should not create user" do
        assert_no_difference 'User.count' do
          post :create, :user => {:name => 'Jean-francois',
                                  :email => 'jf@rubyfrance'}
          response.should render_template('new')
        end
      end
    end

  end
end
```

1 Rspec, BDD ?

2 Rails est agile avant tout

3 La couche modèle de Rails

4 La couche controller de Rails

5 Les routes Rails

6 La couche Vue de Rails

7 les mocks ?

Une simple ligne de route

```
1  ActionController::Routing::Routes.draw do |map|
2    map.resources :users
3  end
```

Doit créer plein de routes

```
describe UsersController do
  describe "route generation" do
    it "should map #index" do
      route_for(:controller => "users",
                :action => "index").should == "/users"
    end

    it "should map #new" do
      route_for(:controller => "users",
                :action => "new").should == "/users/new"
    end

    it "should map #show" do
      route_for(:controller => "users",
                :action => "show", :id => 1).should == "/users/1"
    end

    it "should map #edit" do
      route_for(:controller => "users",
                :action => "edit", :id => 1).should == "/users/1/edit"
    end

    it "should map #update" do
      route_for(:controller => "users",
                :action => "update", :id => 1).should == "/users/1"
    end

    it "should map #destroy" do
      route_for(:controller => "users",
                :action => "destroy", :id => 1).should == "/users/1"
```

1 Rspec, BDD ?

2 Rails est agile avant tout

3 La couche modèle de Rails

4 La couche controller de Rails

5 Les routes Rails

6 La couche Vue de Rails

7 les mocks ?

Une vue d'index

```
1  <h1>Listing users</h1>
2
3  <table>
4    <tr>
5      <th>Name</th>
6      <th>Email</th>
7    </tr>
8
9  <% for user in @users %>
10   <tr>
11     <td><%=h user.name %></td>
12     <td><%=h user.email %></td>
13     <td><%= link_to 'Show', user %></td>
14     <td><%= link_to 'Edit', edit_user_path(user) %></td>
15     <td><%= link_to 'Destroy', user, :confirm => 'Are you sure?',
16                                      :method => :delete %></td>
17   </tr>
18 <% end %>
19 </table>
```

Doit permettre de voir les utilisateurs

```
1   describe "/users/index.html.erb" do
2     include UsersHelper
3
4     before(:each) do
5       assigns[:users] = User.all
6       #There are 2 users in fixtures
7     end
8
9     it "should render list of users" do
10      render "/users/index.html.erb"
11      response.should have_tag("tr>td", User.first.name)
12      response.should have_tag("tr>td", User.first.email)
13    end
14  end
```

1 Rspec, BDD ?

2 Rails est agile avant tout

3 La couche modèle de Rails

4 La couche controller de Rails

5 Les routes Rails

6 La couche Vue de Rails

7 les mocks ?

S'il te plait, dessine moi un mock ?

- mock_model(User)
- Comportement d'un objet ActiveRecord sans access a la BDD
- Possibilite de retourner ce que l'on veux
- Evite de créer une fixture qui gere ce cas la

Doit s'utiliser dans les controllers

```
1   describe UserController do
2     describe "responding to GET show" do
3       before :each do
4         @user = mock_model(User)
5       end
6
7       it "should expose the requested user as @user" do
8         User.should_receive(:find).with("37").and_return(@user)
9         get :show, :id => "37"
10        assigns[:user].should equal(@user)
11      end
12
13      describe "with mime type of xml" do
14
15        it "should render the requested user as xml" do
16          request.env["HTTP_ACCEPT"] = "application/xml"
17          User.should_receive(:find).with("37").and_return(@user)
18          @user.should_receive(:to_xml).and_return("generated XML")
19          get :show, :id => "37"
20          response.body.should == "generated XML"
21        end
22
23      end
24    end
25  end
```

Question ?