Effective Rails Testing Practices

Mike Swieton

atomicobject.com
2007: 16,000 hours
• General testing strategies
  • Integration tests
  • View tests
  • Controller tests
  • Migration tests
• Test at a high level: **Integration tests**
• Test at a low level: **Unit tests**
• **Integration Tests**
  • Every user story
  • Every normal error case
  • Load every page
  • Render every partial
  • Submit every form

• **Unit Tests**
  • Every class
  • Every method
  • Every edge case
  • Every error case
  • Are comprehensive
Story Driven Development
• **Understand** the user story
• Create an **integration test** that represents the story
• **Implement** it starting from the view, letting tests drive your development
• General testing strategies
• Integration tests
• View tests
• Controller tests
• Migration tests
should 'be taken to accept the terms of service if the terms have been updated' do
  now = DateTime.now - 2.seconds
  ensure_larry_has_accepted_current_tos now
  ensure_tony_has_accepted_current_tos now

  with_user(:larry, :password => 'secret') do
    log_in_to_user_dashboard
    logout
  end

  with_user(:tony, :password => 'secret') do
    log_in_to_user_dashboard
    go_to_edit_community_info
    submit_form do |f|
      f.community.terms_of_service = 'these are the new terms'
    end
    assert_response :redirect
    follow_redirect!
  end
end

# ...
should 'be taken to accept the terms of service if the terms have been updated' do
  now = DateTime.now - 2.seconds
  ensure_larry_has_accepted_current_tos now
  ensure_tony_has_accepted_current_tos now

  with_user(:larry, :password => 'secret') do
    log_in_to_user_dashboard
    logout
  end

  with_user(:tony, :password => 'secret') do
    log_in_to_user_dashboard
    go_to_edit_community_info
    submit_form do |f|
      f.community.terms_of_service = 'these are the new terms'
    end
  assert_response :redirect
  follow_redirect!
  # ...
should 'be taken to accept the terms of service if the terms have been updated' do
  now = DateTime.now - 2.seconds
  ensure_larry_has_accepted_current_tos now
  ensure_tony_has_accepted_current_tos now

  with_user(:larry, :password => 'secret') do
    log_in_to_user_dashboard
    logout
  end

  with_user(:tony, :password => 'secret') do
    log_in_to_user_dashboard
    go_to_edit_community_info
    submit_form do |f|
      f.community.terms_of_service = 'these are the new terms'
    end
    assert_response :redirect
    follow_redirect!
  end
end
def log_in
    go_to_login_page
    see_that_we_are_on_login_page
    enter_login_and_password
end

def log_in_to_user_dashboard
    log_in
    see_user_dashboard
end

def see_that_we_are_on_login_page
    assert_select 'form#f_signin'
end

def enter_login_and_password
    submit_form do |form|
        form.user.login = @user.login
        form.user.password = @password
    end
    assert_response :redirect
    # ...

def log_in
    go_to_login_page
    see_that_we_are_on_login_page
    enter_login_and_password
end

def log_in_to_user_dashboard
    log_in
    see_user_dashboard
end

def see_that_we_are_on_login_page
    assert_select 'form#f_signin'
end

def enter_login_and_password
    submit_form do |form|
        form.user.login = @user.login
        form.user.password = @password
    end
    assert_response :redirect
    # ...

FormTestHelper
http://code.google.com/p/form-test-helper/
• General testing strategies
• Integration tests
• View tests
• Controller tests
• Migration tests
• Test every method
• Test all edge cases
• To do this, you need **small methods**
rake stats
<table>
<thead>
<tr>
<th>Name</th>
<th>LOC/M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controllers</td>
<td>8</td>
</tr>
<tr>
<td>Helpers</td>
<td>10</td>
</tr>
<tr>
<td>Models</td>
<td>7</td>
</tr>
<tr>
<td>Libraries</td>
<td>6</td>
</tr>
<tr>
<td>Components</td>
<td>0</td>
</tr>
<tr>
<td>Finders</td>
<td>9</td>
</tr>
<tr>
<td>Presenters</td>
<td>5</td>
</tr>
<tr>
<td>Name</td>
<td>LOC/M</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>Controllers</td>
<td>8</td>
</tr>
<tr>
<td>Helpers</td>
<td>10</td>
</tr>
<tr>
<td>Models</td>
<td>7</td>
</tr>
<tr>
<td>Libraries</td>
<td>6</td>
</tr>
<tr>
<td>Components</td>
<td>0</td>
</tr>
<tr>
<td>Finders</td>
<td>9</td>
</tr>
<tr>
<td>Presenters</td>
<td>5</td>
</tr>
<tr>
<td>Finder specs</td>
<td>14</td>
</tr>
<tr>
<td>Mailer specs</td>
<td>3</td>
</tr>
<tr>
<td>Presenter specs</td>
<td>7</td>
</tr>
<tr>
<td>Model specs</td>
<td>6</td>
</tr>
<tr>
<td>View specs</td>
<td>7</td>
</tr>
<tr>
<td>Controller specs</td>
<td>8</td>
</tr>
<tr>
<td>Helper specs</td>
<td>4</td>
</tr>
<tr>
<td>Library specs</td>
<td>8</td>
</tr>
<tr>
<td>Migration tests</td>
<td>17</td>
</tr>
<tr>
<td>Integration tests</td>
<td>6</td>
</tr>
<tr>
<td>Functional tests</td>
<td>7</td>
</tr>
<tr>
<td>Unit tests</td>
<td>6</td>
</tr>
</tbody>
</table>
The Unit Test Toolbox
Mock objects
• Mock objects
• Mock objects
• Use RSpec
• http://rspec.info/documentation/rails/
Built-in

- Functional tests
- Test views
- Test actions

RSpec

- View specs
  - Test views
- Controller specs
  - Test controllers
Unit Testing Actions
fun_controller.rb

```ruby
def doit
  assigns[:user] = User.find(14)
end
```

fun_controller_spec.rb

```ruby
describe FunController do
  it "does it" do
    user = mock_model(User)
    User.should_receive(:find).with(14).and_return(user)
    get :doit
    assigns[:user].should == user
  end
end
```
**A Controller**

**An Action**

some code do something
work work worksome code do
something work work
worksome code do something
work work worksome code do
something work work work
A Controller

An Action

some code glue glue code

A Helper

code code code

Thing 1

code code code

Thing 2

code code code
Dependency Injection

- Traditional code
  - SomeHelper.new

- With injection
  - SomeHelper provided to constructor
Injection

http://atomicobjectrb.rubyforge.org/
class LoginController < ApplicationController
  inject :landing_page_director,
        :accessors => true

  def authenticate
    if valid_login?
      redirect_to(@landing_page_director.
                 redirect_user(user))
    # ...
  end
class LoginController < ApplicationController
  inject :landing_page_director,
        :accessors => true

  def authenticate
    if valid_login?
      redirect_to(@landing_page_director.redirect_user(user))
    # ...
  end
login_controller.rb

```ruby
class LoginController < ApplicationController
  inject :landing_page_director,
        :accessors => true

  def authenticate
    if valid_login?
      redirect_to(@landing_page_director.redirect_user(user))
      # ...
  end
end
```
login_controller.rb

```
def authenticate
  if valid_login?
    redirect_to(@landing_page_director.redirect_user(user))
    # ...
  end
end
```

```
login_controller_spec.rb

it 'redirects valid user to proper landing page' do
  landing_page_director = create_mock('landing_page_director')
  @controller.landing_page_director = landing_page_director
  landing_page_director.should_receive(:redirect_user).with(@user).
  and_return('http://foo.com')
end
```
login_controller.rb

class LoginController < ApplicationController

  inject :landing_page_director, :accessors = true

  def authenticate
    if valid_login?
      redirect_to(@landing_page_director.redirect_user(user))
    # ...

login_controller_spec.rb

it 'redirects valid user to proper landing page' do

  landing_page_director =
    create_mock('landing_page_director')
  @controller.landing_page_director =
    landing_page_director

  landing_page_director.should_receive(:redirect_user).with(@user).
  and_return('http://foo.com')

  def authenticate
    if valid_login?
      redirect_to(@landing_page_director.redirect_user(user))
    # ...
  end

Sidebar on Fixtures

• Good for integration tests
• Good for “good data”
• Not so good for unit tests
  • Prefer mocks - they’ll run faster too
• Don’t be afraid to insert new records in integration and unit tests for a specific case
Unit Testing Views
hello.rhtml

```html
<div id="username">
  Hi, <%= @user.name %>!
</div>
```

hello_view_spec.rb

```ruby
describe "a view test" do
  it "greets the user" do
    assigns[:user] = stub(
      :name => "Willy J. Stankifier")
    render "hello"
    assert_select "#username",
      "Willy J. Stankifier".to_regex
  end
end
```
with_partial.rhtml

```rhtml
<div id="craphappens">
  <% if @error_message %>
  <%= render :partial => "shared/error",
            :locals => { :message => @error_message } %>
  <% end %>
</div>
```

with_partial_view_spec.rb

```ruby
describe "a view with a partial" do
  it "renders the partial" do
    assigns[:error_message] = "O Snap!"
    template.expect_render(
      :partial => "shared/error",
      :locals => { :message => "O Snap!" })
    render "with_partial"
    assert_select "#craphappens", /O Snap/ end
end
end
```
• General testing strategies
• Integration tests
• View tests
• Controller tests
• Migration tests
• Test them too.
• Seriously.
• Usually developed quickly
• Run against production data
• With no or few of your data validations
• Often perform tricky transformations
• Usually developed quickly
• Run against production data
• With no or few of your data validations
• Often perform tricky transformations
• and you don’t have tests for it?!
They’re not disposable.
MigrationTestHelper

http://migrationtest.rubyforge.org/
Testing All Migrations

http://migrationtest.rubyforge.org/
def see_full_schema
assert_schema do |s|
  s.table :sessions do |t|
    t.column :id, :integer
    t.column :session_id, :string
    t.column :data, :text
    t.column :updated_at, :datetime
  end

  s.table :users do |t|
    t.column :id, :integer
    t.column :login, :string
    t.column :salt, :string
    t.column :password, :string
    t.column :owner_id, :integer
    t.column :owner_type, :string
    t.column :created_at, :datetime
    t.column :updated_at, :datetime
    # ...
  end
Testing One Migration

http://migrationtest.rubyforge.org/
```ruby
def test_migration_munge_the_data
drop_all_tables

migrate :version => 75
# < setup test data >

migrate :version => 76
# < test the results >

migrate :version => 75
# < test the down migration >
end
```

Migrate to any schema version
But I already have a project going!
• RSpec on Rails can coexist with existing tests
• RSpec on Rails can coexist with existing tests
• Injection can be added to new code with zero impact
• RSpec on Rails can coexist with existing tests
• Injection can be added to new code with zero impact
• Migration tests
Put it all together
- Use Injection to help you break code into bite-sized bits.
- Use RSpec to test controllers and views separately.
- Use integration tests to ensure your applications continues to work.
- Test migrations with the MigrationTestHelper.

http://atomicobject.com
• Injection:
  http://rubyforge.org/projects/atomicobjectrb/

• MigrationTestHelper:
  http://migrationtest.rubyforge.org/

• RSpec on Rails
  http://rspec.info/documentation/rails/

• FormTestHelper
  http://code.google.com/p/form-test-helper/

• Shoulda
  http://www.thoughtbot.com/projects/shoulda

http://atomicobject.com