

The Great Test Framework Dance-off

Josh Susser – Pivotal Labs
RailsConf 2008, May 31

*De gustibus non
est disputandum.*

Goals

Compare Test Frameworks

Help You Choose

Method

**What are the
most popular test
frameworks?**

RSpec
test/unit
Shoulda

RailsEnvy poll 4/2/2008

**TDD a modest
Rails application**

use test/spec as
neutral
framework

**convert tests to
three frameworks**

**compare and
evaluate**

The Code

Teldra

600 LOC

6 Models

5 Controllers

At a glance

most differences
are superficial

specialized
support for Rails
is cool

**magic sometimes
has a high cost**

“test”

vs

“spec”

terminology is
mostly arbitrary

perspective

**syntax is a
shibboleth**

Side by Side
by Side

Structure

RSpec

Example Group

describe

```
describe Article do
  describe "Blog list" do
    scenario :blog

    it "should find only posts in reverse chron order" do
      # ...
    end

    it "should find specified number of recent posts" do
      # ...
    end
  end
end

describe "#find_post_by_date_and_slug" do
  scenario :blog

  it "should find the post with slug published on given date" do
    # ...
  end
end
```

test/unit

Class

```
class ArticleTest < ActiveSupport::TestCase
  class BasicTest < ArticleTest
    scenario :basic

    def test_should_return_content_as_body_and_extended
      # ...
    end

    def test_should_return_content_body_if_extended_is_blank
      # ...
    end
  end
end

class BlogTest < ArticleTest
  scenario :blog

  def test_should_find_only_posts_in_reverse_chron_order
    # ...
  end
end
```

Shoulda
context

```
class ArticleTest < Test::Unit::TestCase
  scenario :blog

  context "Blog list" do
    should "find only posts in reverse chron order" do
      # ...
    end

    should "find specified number of recent posts" do
      # ...
    end
  end

  context "Finding posts with #find_post_by_date_and_slug" do
    should "find the post with slug published on given date" do
      # ...
    end
  end
end
```

```
class Admin::PanelControllerTest < ActionController::TestCase
  scenario :blog

  context "when logged out" do
    setup { logout }
    context "index" do
      setup { get :index }
      should_redirect_to "login_url"
    end
  end

  context "when logged in as admin" do
    setup { login_as :admin }
    context "index" do
      setup { get :index }
      should_render_template "index"
    end
  end
end
```

Test Case

`test/unit`

`method`

`def test_thing`

```
def test_should_find_the_page_with_slug
  about = articles(:about)
  page = Article.find_page_by_slug(about.slug)
  assert_equal about, page
end
```

RSpec

Example

it “should...” do

```
it "should find the page with slug" do
  about = articles(:about)
  page = Article.find_page_by_slug(about.slug)
  page.should == about
end
```

Shoulda

test

should “...” do

```
should "find the page with the slug" do
  about = articles(:about)
  page = Article.find_page_by_slug(about.slug)
  assert_equal about, page
end
```

Assertion

```
assert_not_nil assigns(:user)  
assigns(:user).should_not be_nil
```

```
assert_equal 2, articles.size  
articles.size.should == 2  
articles.should have(2).items
```

```
assert_template "show"  
response.should render_template("show")  
should_render_template "show"
```

Sharing

```
describe "an authenticated controller", :shared => true do
  it "should redirect to the the login page" do
    # ...
  end
end
```

```
describe Admin::CommentsController do
  it_should_behave_like "an authenticated controller"
end
```

```
module AuthenticatedRestfulTests
  def test_should_redirect_to_the_the_login_page
    # ...
  end
end

class Admin::ArticlesControllerTest < ActionController::TestCase
  include AuthenticatedRestfulTests
end
```

```
def self.should_authenticate_restful_actions
  context "when logged out" do
    setup { logout }
    should "redirect any restful action to login page" do
      # ...
    end
  end
end
```

```
class Admin::ArticlesControllerTest < ActionController::TestCase
  should_authenticate_restful_actions
end
```

Extending

```
def assert_sorted(actual, message=nil, &block)
  expected = actual.sort(&block)
  assert_equal expected, actual, "Order is wrong:"
end
```

```
assert_sorted(tags) { |a,b| a.name <=> b.name }
```

```
module Spec
  module Matchers
    def be_sorted(&block)
      BeSorted.new(&block)
    end

    class BeSorted
      def initialize(&block)
        @sort_block = block
      end

      def matches?(actual)
        @actual = actual
        @expected = @actual.sort(&@sort_block)
        @actual == @expected
      end

      def failure_message
        "expected collection in order:\n" +
        @expected.inspect +
        "\nbut got order:\n" +
        @actual.inspect
      end

      def negative_failure_message
        "expected collection not to be in order:\n" +
        @actual.inspect
      end
    end
  end
end

tags.should be_sorted { |a,b| a.name <=> b.name }
```

Getting Feedback

test/unit

.....F.....
.....

RSpec

Article

Article Blog list

should find only posts in reverse chron order

should find specified number of recent posts

Article#find_post_by_date_and_slug

should find the post with slug published on given date

should raise error when called with bogus date

Article#find_page_by_slug

should find the page with slug

Article Published post.post_path_params

should return array of params for creating url

should raise when called on a non-post article

Ease of Use

Easy to Read

Terse

but not cryptic

LOC

test/unit	616
RSpec	560*
Shoulda	690

Easy to
Understand

**accessible
abstractions
and syntax**

Easy to Write

```
# RSpec
describe Article do
  describe "Blog list" do

# Shoulda
class ArticleTest < Test::Unit::TestCase
  context "Blog list" do

# test/unit
class ArticleTest < ActiveSupport::TestCase
  class BasicTest < ArticleTest
```

Easy to Extend

:) assert_custom

:(be_custom

8) should_custom

Easy to Refactor

**What matters
most to you?**

(Performance)

Fixture Scenarios

:(nested contexts

cartesian product

=> slow

Opinionated

FTW

Helps direct your
test strategy

test/unit

so neutral it doesn't

give much help

RSpec

nested describes

are great...

but don't constrain

enough

Shoulda

macros strongly

encourage an

effective testing style

Mix and Match

Structure
Test Cases
Assertions

IMHO

Structure:

describe

(it's really a class)

Test Case:

it / should

Assertion:

assert_whatever

Extend:

assert_custom

Refactor:

should_dry_it_up

github.com/joshsusser/teldra_prime

(real Teldra release RSN)



Creative Commons Attribution-Noncommercial 3.0 Unported License