

# Testing in Ruby

Joseph Wilk

<http://www.joesniff.co.uk>



# Examples

- Look at **testing tools** available for **Ruby**
- Run some examples taken from

```
git clone git://github.com/eshopworks/rboss-gem.git
```

- Get the code and **try for yourself**



# What to test

- Objects
- System
- Complexity
- Tests!



# What to test

Everything

- Objects
- System
- Complexity
- Tests!



# Rspec

## Quick Intro:

RSpec is a **Behaviour Driven Development** framework for Ruby.

## Two frameworks

- Story
- Specs

Writing and executing examples of how your Ruby application should **behave**



# Objects



# Spec's (RSpec)

- A Spec Framework for describing **behaviour** at the **object level**

describe 'context'  
it 'should behave like this'

- Mock/Stub to help isolated the System Under Test
  - Flexmock, RR (Test double framework), Mocha

```
require File.dirname(__FILE__) + '/../spec_helper'

describe Boss::Api do

  def mock_http_response(stubs={})
    mock('http_response', { :body => '{"ysearchresponse":{}}', :code =>
"200"}.merge(stubs))
  end

  before(:each) do
    @api = Boss::Api.new( appid = 'test' )
    @api.endpoint = 'http://www.example.com/'
  end

  describe "responding to spelling search" do

    it "should make a spelling request to yahoo service" do
      Net::HTTP.should_receive(:get_response).and_return{ mock_http_response }

      @api.search_spelling("girafes")
    end

    it "should build the spelling objects" do
      Net::HTTP.stub!(:get_response).and_return{ mock_http_response }
      Boss::ResultFactory.should_receive(:build).with('{"ysearchresponse":{}}')

      @api.search_spelling("girafes")
    end

  end

end
```



System





# Cucumber

Re-implementation of RSpec's story framework, based on Treetop by Aslak Hellesoy

- **Features** rather than stories
- **FIT** (functional integration tests) tables
- **Internalisation** support
- Better Error reporting



Feature: Web Search

In order to get search results

As a API user

I want to query yahoo boss

Scenario: Search web

Given a valid API key

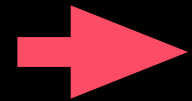
When I do a 'web' search for 'monkeys'

Then I will receive search results

And I will be able to see the total hits

type	term	
web	monkey	
images	monkey	
news	monkey	
spell	<u>girafe</u>	





## Feature: Web Search

In order to get search results

As a API user

I want to query yahoo boss

Scenario: Search web

Given a valid API key

When I do a 'web' search for 'monkeys'

Then I will receive search results

And I will be able to see the total hits

type	term	
web	monkey	
images	monkey	
news	monkey	
spell	<u>girafe</u>	



Feature  
not story →

## Feature: Web Search

In order to get search results

As a API user

I want to query yahoo boss

### Scenario: Search web

Given a valid API key

When I do a 'web' search for 'monkeys'

Then I will receive search results

And I will be able to see the total hits

type	term	
web	monkey	
images	monkey	
news	monkey	
spell	<u>girafe</u>	



Feature  
not story → Feature: Web Search

→ In order to get search results  
As a API user  
I want to query yahoo boss

Scenario: Search web

Given a valid API key

When I do a 'web' search for 'monkeys'

Then I will receive search results

And I will be able to see the total hits

type	term	
web	monkey	
images	monkey	
news	monkey	
spell	<u>girafe</u>	



Feature not story → Feature: Web Search

Value first → In order to get search results  
As a API user  
I want to query yahoo boss

Scenario: Search web

Given a valid API key

When I do a 'web' search for 'monkeys'

Then I will receive search results

And I will be able to see the total hits

type	term	
web	monkey	
images	monkey	
news	monkey	
spell	<u>girafe</u>	



Feature not story → Feature: Web Search

Value first → In order to get search results  
As a API user  
I want to query yahoo boss

Scenario: Search web

Given a valid API key

When I do a 'web' search for 'monkeys'

Then I will receive search results

And I will be able to see the total hits

→

type	term	
web	monkey	
images	monkey	
news	monkey	
spell	<u>girafe</u>	



Feature not story → Feature: Web Search

Value first → In order to get search results  
As a API user  
I want to query yahoo boss

Scenario: Search web

Given a valid API key

When I do a 'web' search for 'monkeys'

Then I will receive search results

And I will be able to see the total hits

FIT table →

type	term	
web	monkey	
images	monkey	
news	monkey	
spell	<u>girafe</u>	



Feature not story → Feature: Web Search

Value first → In order to get search results  
As a API user  
I want to query yahoo boss

Scenario: Search web

Given a valid API key

When I do a 'web' search for 'monkeys'

Then I will receive search results

And I will be able to see the total hits

FIT table →

type	term	
web	monkey	
images	monkey	
→   news	monkey	
spell	<u>girafe</u>	



Feature not story → Feature: Web Search

Value first → In order to get search results  
As a API user  
I want to query yahoo boss

Scenario: Search web

Given a valid API key

When I do a 'web' search for 'monkeys'

Then I will receive search results

And I will be able to see the total hits

FIT table →

type	term
------	------

web	monkey
-----	--------

images	monkey
--------	--------

FIT values →

news	monkey
------	--------

spell	<u>girafe</u>
-------	---------------



> rake features



## Feature: Web Search

In order to get search results

As a API user

I want to query yahoo boss

## Scenario: Search web

Given a valid API key

When I do a 'web' search for 'monkeys'

Then I will receive search results

And I will be able to see the total hits

|type|term|

|web|monkey|

|images|monkey|

|news|monkey|

|spell|girafe|

20 steps passed

50 2f6b2 bazz6q

|2b6j|g1r4t6|

|u6m2|monkey|



# Webrat

- Ruby Acceptance Testing for Web applications
- Simulates the DOM
- Does this all **in memory**
  - FAST!
- Supports
  - Form interaction
  - Clicking links
  - Post/Gets
- **No** handling for **Javascript**
- Gets **confused** with **complex HTML**



# Testing :through => “browser”

- Watir - <http://wtr.rubyforge.org/>
  - Ruby powered
- Selenium - <http://selenium.rubyforge.org/>
  - Java server (Selenium-rc)

Slow compared to Webrat

Supports Javascript and complex HTML



# Complexity



# Flog

Flog essentially scores a **complexity metric** based on:

- Assignments
- Branches
- Calls

Excessive usage of these can be an indicator of too complex code.

Things at the **top of the list** are worth taking a **closer look** at.



>flog lib



ResultFactory#build: (54.7)

21.8: branch  
14.3: assignment  
12.7: new  
10.0: <<  
7.1: []  
4.1: each  
4.0: has\_key?  
2.2: discover\_search\_type  
2.1: set\_instance\_variable  
2.0: raise  
2.0: include?  
1.9: parse

Api#search\_boss: (39.7)

12.1: branch  
11.5: assignment  
5.6: raise  
3.2: format  
3.0: format?  
2.8: new  
2.6: empty?  
1.7: build  
1.6: body  
1.6: join  
1.6: parse\_error  
1.5: count  
1.5: build\_request\_url  
1.4: include?  
1.4: yield  
1.3: parse  
1.3: block\_given?  
1.3: get\_response  
1.3: >  
1.3: code

main#none: (26.0)

15.4: require  
4.4: dirname  
3.2: branch  
2.4: include?  
1.4: expand\_path  
1.2: unshift  
1.1: assignment  
1.0: each

Config#to\_url: (14.9)

7.2: assignment  
3.6: []  
2.6: inject  
1.7: to\_s  
1.5: escape  
1.5: +  
1.5: marshal\_dump  
1.3: branch  
0.5: lit\_fixnum



# Testing tests



# Rcov

Examine which parts of your  
code are  
not covered by your tests.

- Generates pretty HTML reports
- Ensure test coverage before commits

```
>rake verify_rcov
```

```
> Coverage: 99.7% (threshold: 99.7%)
```



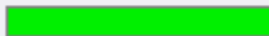
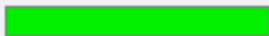


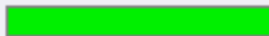
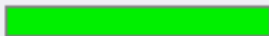
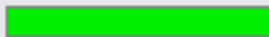
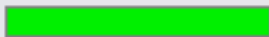




















> rake spec

> cd coverage



## C0 code coverage information

Generated on Fri Sep 05 12:43:20 +0100 2008 with [rcov 0.8.1.2](#)

Name	Total lines	Lines of code	Total coverage	Code coverage
TOTAL	292	214	99.7% 	99.5% 
<a href="#">lib/boss.rb</a>	23	18	95.7% 	94.4% 
<a href="#">lib/boss/api.rb</a>	93	69	100.0% 	100.0% 
<a href="#">lib/boss/config.rb</a>	22	14	100.0% 	100.0% 
<a href="#">lib/boss/result.rb</a>	1	1	100.0% 	100.0% 
<a href="#">lib/boss/result/base.rb</a>	15	10	100.0% 	100.0% 
<a href="#">lib/boss/result/image.rb</a>	8	6	100.0% 	100.0% 
<a href="#">lib/boss/result/news.rb</a>	8	6	100.0% 	100.0% 
<a href="#">lib/boss/result/spell.rb</a>	8	6	100.0% 	100.0% 
<a href="#">lib/boss/result/web.rb</a>	8	6	100.0% 	100.0% 
<a href="#">lib/boss/result_collection.rb</a>	29	22	100.0% 	100.0% 
<a href="#">lib/boss/result_factory.rb</a>	61	43	100.0% 	100.0% 
<a href="#">lib/boss/version.rb</a>	9	8	100.0% 	100.0% 
<a href="#">spec/spec_helper.rb</a>	7	5	100.0% 	100.0% 



Code reported as executed by Ruby looks like  
this...

and this: this line is also marked as  
covered.

Lines considered as run by rcov, but  
not reported by Ruby, look like this,  
and

this: these lines were inferred by rcov (using simple heuristics).

Finally, here's a line marked as not  
executed.

Name	Total lines	Lines of code	Total coverage	Code coverage
<a href="#">lib/boss.rb</a>	23	18	95.7% <div><div></div></div>	94.4% <div><div></div></div>

```
1
  $:.unshift(File.dirname(__FILE__)) unless
2 $:.include?(File.dirname(__FILE__)) ||
  $:.include?(File.expand_path(File.dirname(__FILE__)))
3
4 require 'boss/api'
5 require 'boss/config'
6 require 'boss/result'
7 require 'boss/result_collection'
8 require 'boss/result_factory'
9
9 require 'boss/version'
```

```
10
11 module Boss
12 YAHOO_VERSION = 1
13
14 module SearchService
15 %w[web images news spelling].each { |e|
16   const_set(e.upcase, e) }
17 end
```

```
18 end
19   const_set(e.upcase, e) }
20 %w[web images news spelling].each { |e|
21   module SearchService
22
23 YAHOO_VERSION = 1
24 module Boss
```



> rake spec



# Spec Profile (Rspec)

Profiling enabled.

.....P.....

Top 10 slowest examples:

0.0343130 Boss::Api failed search should extract error from xml on failed search

0.0083660 Boss::Api failed search should raise error on failed search

0.0053660 Boss::Api configuring search should allow configuring through block

0.0052650 Boss::Api searching terms should encode invalid characters

0.0045870 Boss::Api responding to spelling search should build the spelling objects

0.0044650 Boss::Api configuring search should allow configuring through hash

0.0044360 Boss::Api responding to web search should build the web objects

0.0044270 Boss::Api responding to image search should build the image objects

0.0044130 Boss::Api responding to news search should build the news objects

0.0042030 Boss::Api formats should not return any objects when format is 'json'



# Heckle

## Hurt your tests

1. Your tests should pass.
2. Break your code.
3. Now they should fail.

Mutates the code.

Be warned! Can lead to infinite loops and general craziness.

if becomes unless

calls get replaced

numbers get changed,

assignments get changed

and more...



```
>spec spec/boss/  
result_collection_spec.  
rb --heckle  
Boss::ResultCollection
```



\*\*\*\*\*

\*\*\* Boss::ResultCollection#set\_instance\_variable loaded with 2 possible mutations

\*\*\*\*\*

2 mutations remaining...

1 mutations remaining...

The following mutations didn't cause test failures:

Binary files original and mutation differ

--- original

+++ mutation

```
def set_instance_variable(name, value)
  instance_variable_set("@#{name}", value)
- instance_eval("def #{name}\n @#{name}\n end")
+ instance_eval("def #{name}\n @#{name}hz")
end
```

\*\*\*\*\*

\*\*\* Boss::ResultCollection#<< loaded with 1 possible mutations

\*\*\*\*\*

1 mutations remaining...

The following mutations didn't cause test failures:

--- original

+++ mutation

```
def <<(element)
- (@results << element)
+ # do nothing
end
```



# Future

- Johnson wraps JavaScript in a loving Ruby embrace.

Webrat + Johnson?

- Cucumber merging with Rspec?



# Questions?