TESTING WITH RSPEC

Introduction

• LEVEL 1 •

RSpec

- Popular Ruby testing framework
- Thriving community
- Less Ruby-like, natural syntax
- Well-formatted output

David Chelimsky

Behavior-Driven Dev

- Describe your application's behavior
  - Can be done with other frameworks
  - BDD is not required
  - But encouraged by the RSpec syntax
  - And RSpec makes it elegant and readable

Installing RSpec

```
$ gem install rspec
...
Successfully installed rspec-core
Successfully installed rspec-expectations
Successfully installed rspec-mocks
Successfully installed rspec
4 gems installed
```

```
$ rspec --init
  create    spec/spec_helper.rb
  create    .rspec
```

← in your project directory

more about configuration in level 2

Describe block

Our project source will live in /lib/zombie.rb

but lets write a test first

spec/lib/zombie_spec.rb ← <name_of_spec>_spec.rb

```ruby
require "spec_helper"
describe "A Zombie" do
  # your examples (tests) go here
end
```

Describe it

spec/lib/zombie_spec.rb

```
require "spec_helper"
describe "A Zombie" do
  it "is named Ash"
end
```

name of the example

examples are declared using the "it" method

Pending

spec/lib/zombie_spec.rb

```
require "spec_helper"
describe "A Zombie" do
  it "is named Ash"
end
```

```
$ rspec spec/lib/zombie_spec.rb
*
Pending:
  Zombie is named "Ash"
    # Not yet implemented
      # ./spec/lib/zombie_spec.rb:17
Finished in 0.00028 seconds
1 example, 0 failures, 1 pending
```

Pending

LEVEL 1 - INTRODUCTION

Describe Class

spec/lib/zombie_spec.rb

```
require "spec_helper"
describe Zombie do
  it "is named Ash"
end
```

class we want to test

```
$ rspec spec/lib/zombie_spec.rb

zombie_spec.rb:16:in `<top (required)>':
uninitialized constant Zombie (NameError)
```

Failing

we don't have a Zombie class yet

Creating the class

spec/lib/zombie_spec.rb

```ruby
require "spec_helper"
require "zombie"

describe Zombie do
  it "is named Ash"
end
```

lib/zombie.rb

```ruby
class Zombie

end
```

```
$ rspec spec/lib/zombie_spec.rb
*
Pending:
  Zombie is named "Ash"
    # Not yet implemented
      # ./spec/lib/zombie_spec.rb:17
Finished in 0.00028 seconds
1 example, 0 failures, 1 pending
```

Pending

Expectations

spec/lib/zombie_spec.rb

```
describe Zombie do
  it "is named Ash" do
    zombie = Zombie.new
    zombie.name.should == 'Ash'
  end
end
```

expectation

Test::Unit

```
assert_equal 'Ash', zombie.name
```

- This is how you ‘assert’ in RSpec
- Assertions are called ‘expectations’
- They read more like plain English

Test properly fails

spec/lib/zombie_spec.rb

```
describe Zombie do
  it "is named Ash" do
    zombie = Zombie.new
    zombie.name.should == 'Ash'
  end
end
```

```
$ rspec spec/lib/zombie_spec.rb

1) Zombie is named "Ash"
     Failure/Error: zombie.name.should == 'Ash'
     NoMethodError:
       undefined method `name' for #<Zombie>

Finished in 0.00125 seconds
1 example, 1 failure
```

Make it pass

spec/lib/zombie_spec.rb

```
describe Zombie do
  it "is named Ash" do
    zombie = Zombie.new
    zombie.name.should == 'Ash'
  end
end
```

lib/zombie.rb

```
class Zombie
  attr_accessor :name

  def initialize
    @name = 'Ash'
  end
end
```

```
$ rspec spec/lib/zombie_spec.rb
.

Finished in 0.00047 seconds
1 example, 0 failures
```

Passed!

Another expectation

spec/lib/zombie_spec.rb

```
describe Zombie do

  ...
  it "has no brains" do
    zombie = Zombie.new
    zombie.brains.should < 1
  end
end
```

matcher

modifier

```
$ rspec spec/lib/zombie_spec.rb

1) Zombie is named "Ash"
     Failure/Error: zombie.brains.should < 1
     NoMethodError:
       undefined method `brains' for #<Zombie>

Finished in 0.00125 seconds
1 example, 1 failure
```

TESTING WITH RSPEC

Make it pass

spec/lib/zombie_spec.rb

```ruby
describe Zombie do

  ...
  it "has no brains" do
    zombie = Zombie.new
    zombie.brains.should < 1
  end
end
```

lib/zombie.rb

```ruby
class Zombie
  attr_accessor :name, :brains

  def initialize
    @name = 'Ash'
    @brains = 0
  end
end
```

```
$ rspec spec/lib/zombie_spec.rb
 ..

 Finished in 0.00045 seconds
 2 example, 0 failures
```

Passed!

Other matchers

```
zombie.name.should == 'Ash'
```

```
zombie.alive.should == false
```

```
zombie.alive.should be_false
```

```
zombie.rotting.should == true
```

```
zombie.alive.should be_true
```

```
zombie.height.should > 5
```

```
zombie.brains.should be < 1
```

```
zombie.height.should >= 5
```

```
zombie.height.should < 5
```

```
zombie.height.should_not == 5
```

Testing a predicate

/lib/zombie.rb

```
class Zombie
  def hungry?
    true            predicate
  end
end
```

/spec/lib/zombie_spec.rb

```
describe Zombie do
  it 'is hungry' do
    zombie = Zombie.new
    zombie.hungry?.should == true
  end
end                could read better
```

Predicate 'be'

/spec/lib/zombie_spec.rb

```
describe Zombie do
  it 'is hungry' do
    zombie = Zombie.new
    zombie.hungry?.should == true
  end
end
```

```
zombie.hungry?.should be_true
```

```
zombie.should be_hungry
```

predicate matcher

Test passes

/spec/lib/zombie_spec.rb

```
describe Zombie do
  it 'is hungry' do
    zombie = Zombie.new
    zombie.should be_hungry
  end
end
```

```
$ rspec spec/lib/zombie_spec.rb
 .

 Finished in 0.00045 seconds
 1 example, 0 failures
```

Still passing!

To mark as pending

```
it "is named Ash"
```

to-do list

```
xit "is named Ash" do
  ...
end
```

Useful for Debugging

```
it "is named Ash" do
  pending
  ...
end
```