TESTING WITH RSPEC

Configuration & Matchers

• LEVEL 2 •

Installing RSpec

```
$ gem install rspec
Fetching: rspec-core.gem (100%)
Fetching: rspec-expectations.gem (100%)
Fetching: rspec-mocks.gem (100%)
Fetching: rspec.gem (100%)
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

Inside Rails

```
group :development, :test do
  gem 'rspec-rails'
end
```

in your Gemfile

```
$ bundle install
...
Installing rspec-core
Installing rspec-expectations
Installing rspec-mocks
Installing rspec
Installing rspec-rails

$ rails generate rspec:install
  create .rspec
  create spec/spec_helper.rb
```

not just rspec core

different with Rails

Configuration

spec/spec_helper.rb

```
Dir[Rails.root.join("spec/support/**/*.rb")].each {|f| require f}
...
RSpec.configure do |config|
  config.mock_with :mocha
  ...
end
```

requires all helper files within spec/support

allows you to change the default mocking framework

Output

```
$ rspec --color spec/models/zombie_spec.rb
.
Finished in 0.00176 seconds
1 examples, 0 failures
```

```
$ rspec --color --format documentation spec/models/zombie_spec.rb

Zombie
  is invalid without a name
.
Finished in 0.00052 seconds
1 examples, 0 failures
```

.rspec

```
--color
--format documentation
```

makes it a default

Running specs

```
$ rspec
```

running all the '_spec.rb' files within /spec

```
$ rspec spec/models/
```

running a specific directory

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

running a specific test

```
$ rspec spec/models/zombie_spec.rb:4
```

running a specific line

Model spec

spec/models/zombie_spec.rb

```ruby
require 'spec_helper'
describe Zombie do
  it 'is invalid without a name' do
    zombie = Zombie.new
    zombie.should_not be_valid
  end
end
```

predicate matcher

app/models/zombie.rb

```ruby
class Zombie < ActiveRecord::Base
  validates :name, presence: true
end
```

```
$ rspec spec/models/zombie_spec.rb
.
Finished in 0.00176 seconds
1 examples, 0 failures
```

Matchers: match

spec/models/zombie_spec.rb

```
describe Zombie do
  it "has a name that matches 'Ash Clone'" do
    zombie = Zombie.new(name: "Ash Clone 1")
    zombie.name.should match(/Ash Clone \d/)
  end
end
```

takes a regular expression

Matchers: include

spec/models/zombie_spec.rb

```ruby
describe Zombie do
  it 'include tweets' do
    tweet1 = Tweet.new(status: 'Uuuuunhhhhh')
    tweet2 = Tweet.new(status: 'Arrrrgggg')
    zombie = Zombie.new(name: 'Ash', tweets: [tweet1, tweet2])
    zombie.tweets.should include(tweet1)
    zombie.tweets.should include(tweet2)
  end
end
```

matching on an array

is this Tweet inside tweets?

Matchers: have

spec/models/zombie_spec.rb

```rb
describe Zombie do
  it 'starts with two weapons' do
    zombie = Zombie.new(name: 'Ash')
    zombie.weapons.count.should == 2
  end
end
```

reads much better

```rb
describe Zombie do
  it 'starts with two weapons' do
    zombie = Zombie.new(name: 'Ash')
    zombie.should have(2).weapons
  end
end
```

these will work too

```rb
have(n)
have_at_least(n)
have_at_most(n)
```


Matchers: change

spec/models/zombie_spec.rb

```ruby
describe Zombie do
  it 'changes the number of Zombies' do
    zombie = Zombie.new(name: 'Ash')
    expect { zombie.save }.to change { Zombie.count }.by(1)
  end
end
```

runs before and after expect

results are compared

give these a shot

```ruby
by(n)
from(n)
to(n)
```

you can chain them

```ruby
.from(1).to(5)
```


raise_error

spec/models/zombie_spec.rb

```ruby
describe Zombie do
  it 'raises an error if saved without a name' do
    zombie = Zombie.new
    expect { zombie.save! }.to raise_error(
      ActiveRecord::RecordInvalid
    )
  end
end
```

optionally pass in an exception

```
to
not_to
to_not
```

these modifiers also work

More matchers

```
respond_to(:<method_name>)
```

```
be_within(<range>).of(<expected>)
```

```
exist
```

```
satisfy { <block> }
```

```
be_kind_of(<class>)
```

```
be_an_instance_of(<class>)
```


More matchers

```
@zombie.should respond_to(:hungry?)
```

```
@width.should be_within(0.1).of(33.3)
```

```
@zombie.should exist
```

```
@zombie.should satisfy { |zombie| zombie.hungry? }
```

```
@hungry_zombie.should be_kind_of(Zombie)
```

HungryZombie < Zombie

```
@status.should be_an_instance_of(String)
```