

# 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

```
Successfully installed rspec-core
Successfully installed rspec-expectations
Successfully installed rspec-mocks
Successfully installed rspec
4 gems installed
```

**\$** gem install rspec

```
$ rspec --init in your project directory create spec/spec_helper.rb create .rspec
```

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

require "spec\_helper"
describe "A Zombie" do
 # your examples (tests) go here
end

<name\_of\_spec>\_spec.rb



#### 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



#### 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



### Describe Class

spec/lib/zombie\_spec.rb

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

class we want to test

we don't have a Zombie class yet



# Creating the class

```
require "spec_helper"
require "zombie"

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

lib/zombie.rb

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
```



# Expectations

- 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





#### it Make

spec/lib/zombie\_spec.rb

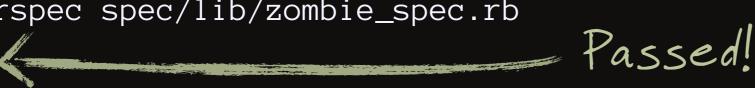
end

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



# another expectation

spec/lib/zombie\_spec.rb

describe Zombie do

• • •

it "has no brains" do

zombie = Zombie.new

Zombie.brains.should < 1</pre>

end

end

matcher

modifier

\$ rspec spec/lib/zombie\_spec.rb

1) Zombie is named "Ash"

Failure/Error: zombie.brains.should < 1</pre>

NoMethodError:

undefined method `brains' for #<Zombie>

Finished in 0.00125 seconds 1 example, 1 failure





#### it Make

spec/lib/zombie\_spec.rb describe Zombie do it "has no brains" do zombie = Zombie.new zombie.brains.should < 1 end end

```
lib/zombie.rb
class Zombie
  attr_accessor :name, :brains
  def initialize
    @name = 'Ash'
    @brains = 0
  end
end
```

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

Finished in 0.00045 seconds 2 example, 0 failures





#### Other matcher's

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</pre>

zombie.height.should >= 5

zombie.height.should < 5</pre>

zombie.height.should\_not == 5



### Testing a predicate

/lib/zombie.rb

```
class Zombie

def hungry?

true

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



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

Still passing!



Finished in 0.00045 seconds 1 example, 0 failures



# To mark as pending

to-do list it "is named Ash" xit "is named Ash" do Useful for Debugging end it "is named Ash" do pending end

