Deliverable

Upload these 5 files to moodle

1. zombie1.rb (slide 21)
2. zombie_spec1.rb (slide 21)
3. zombie2.rb (slide 39)
4. zombie_spec2.rb (slide 39)
5. zombie3.rb (slide 41)
6. Read through "slides at home" [by your next midterm]. Information covered in the slides might show up on your next midterm

These slides are taken from codeschool.com


Introduction
• LEVEL 1 •

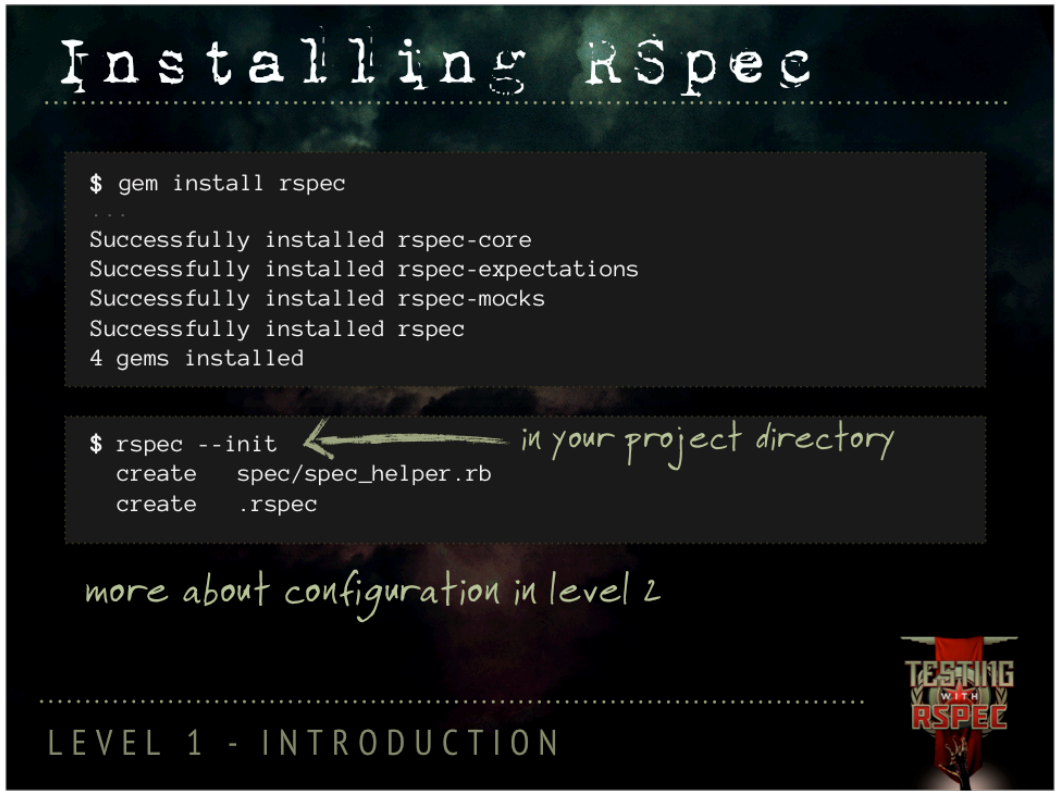
RSpec

- 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



Using rspec without rails

1. Create a directory in rails_project directory and call it lab4-1
2. cd lab4-1
3. Create a directory called “lib”
4. Continue with slide
5. *sudo * gem install rspec

1. Create “lib” directory inside spec
2. Create file zombie_spec.rb in spec/lib
3. Type in code from slide

Describe it

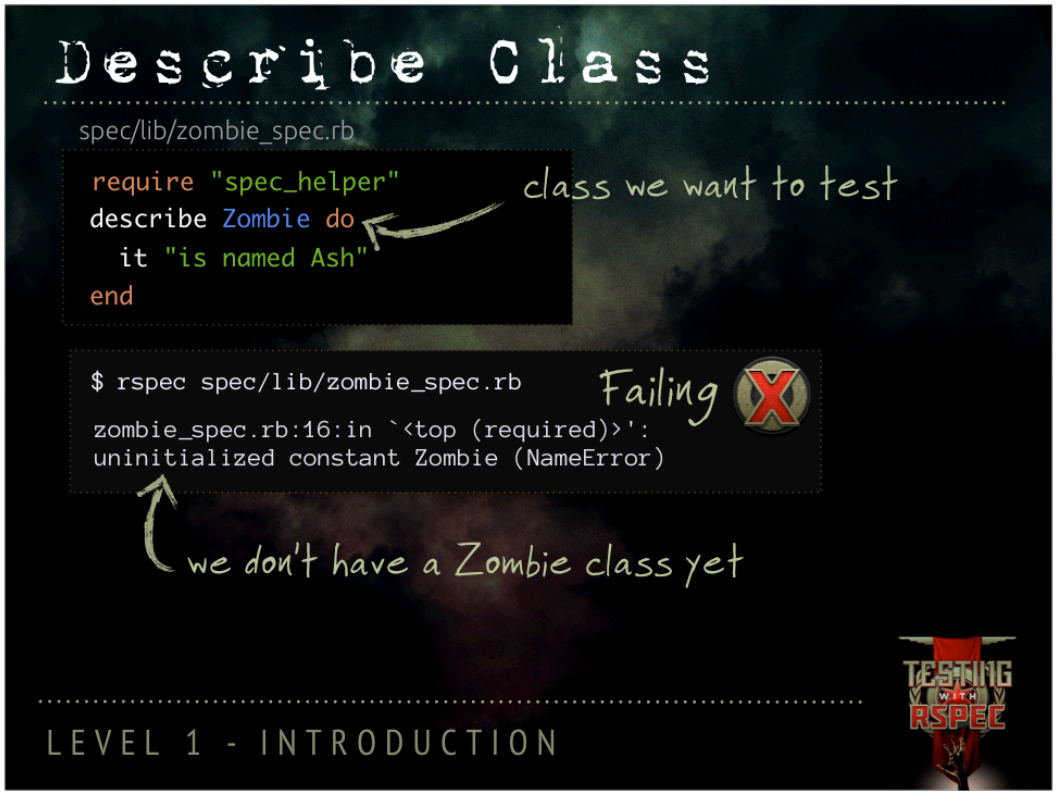
spec/lib/zombie_spec.rb

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

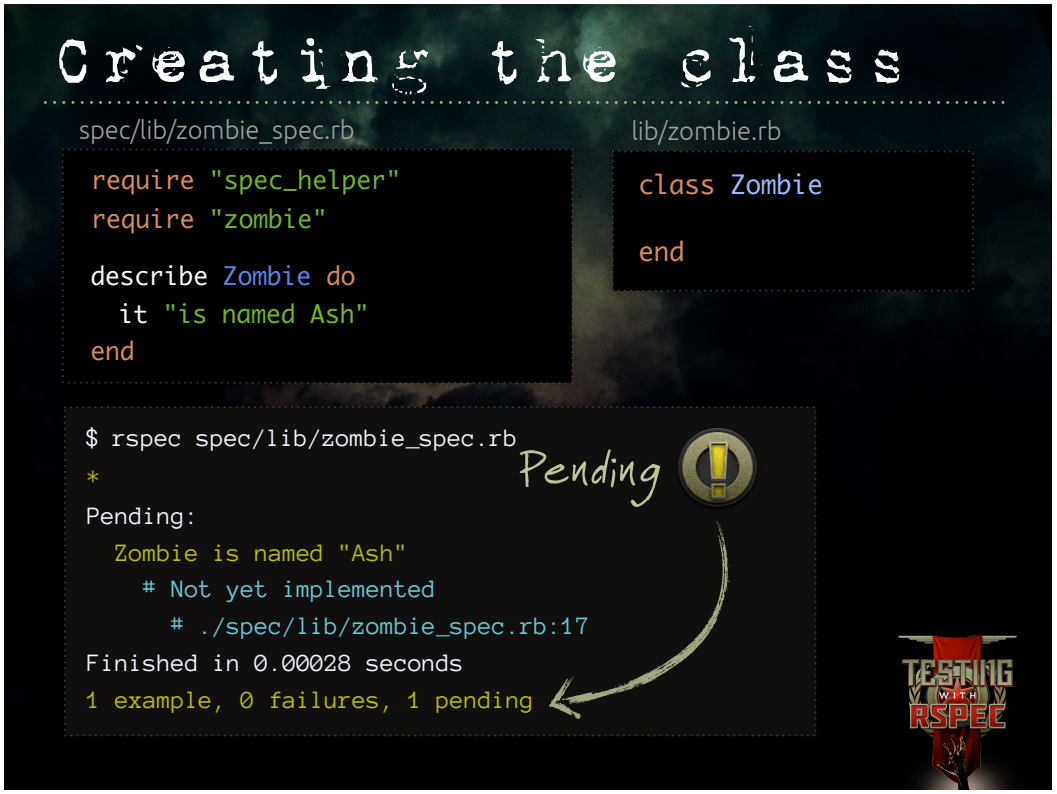
name of the example

examples are declared using the "it" method

1. Run rspec on zombie_spec.rb
2. Should be “pending”



1. Run rspec on zombie_spec.rb
2. See it fail



1. Create zombie.rb in lab4-1/lib
2. Type in the code in the top right corner into zombie.rb
3. Add require ‘zombie’ into zombie_spec.rb
4. Run rspec

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
```

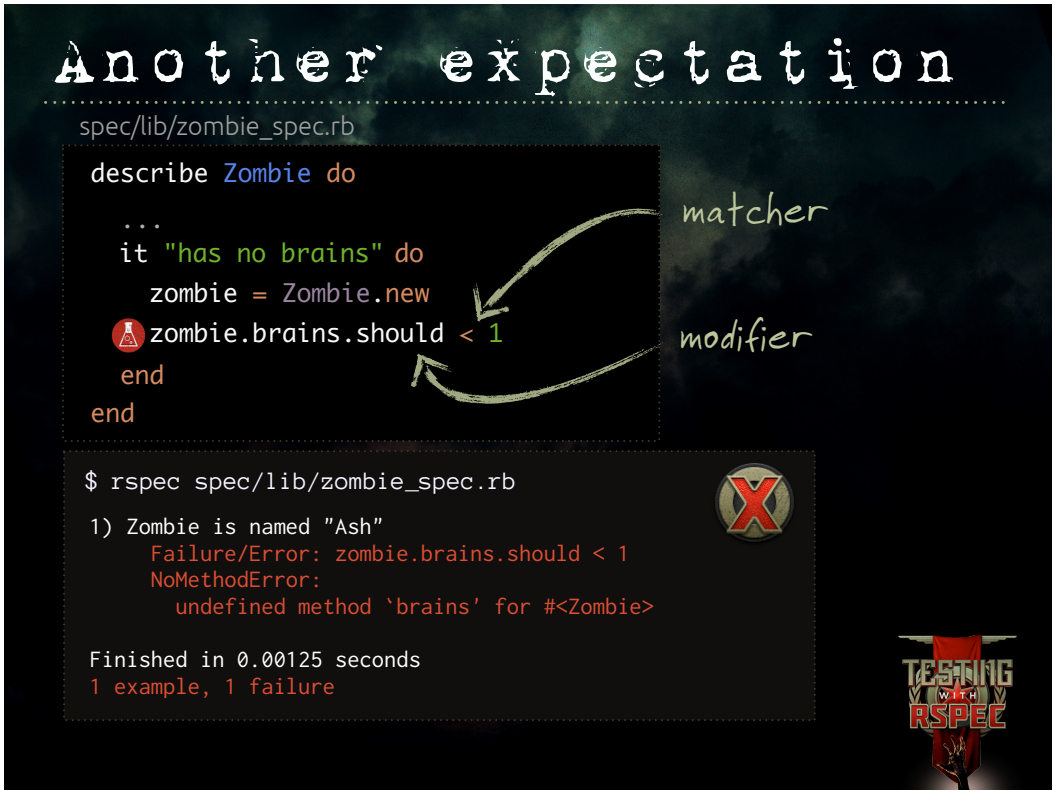
```
$ rspec spec/lib/zombie_spec.rb

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
```

1. Type code into appropriate file
2. Run rspec
3. See if fail

1. Add lines into zombie.rb
2. Run rspec
3. See it pass

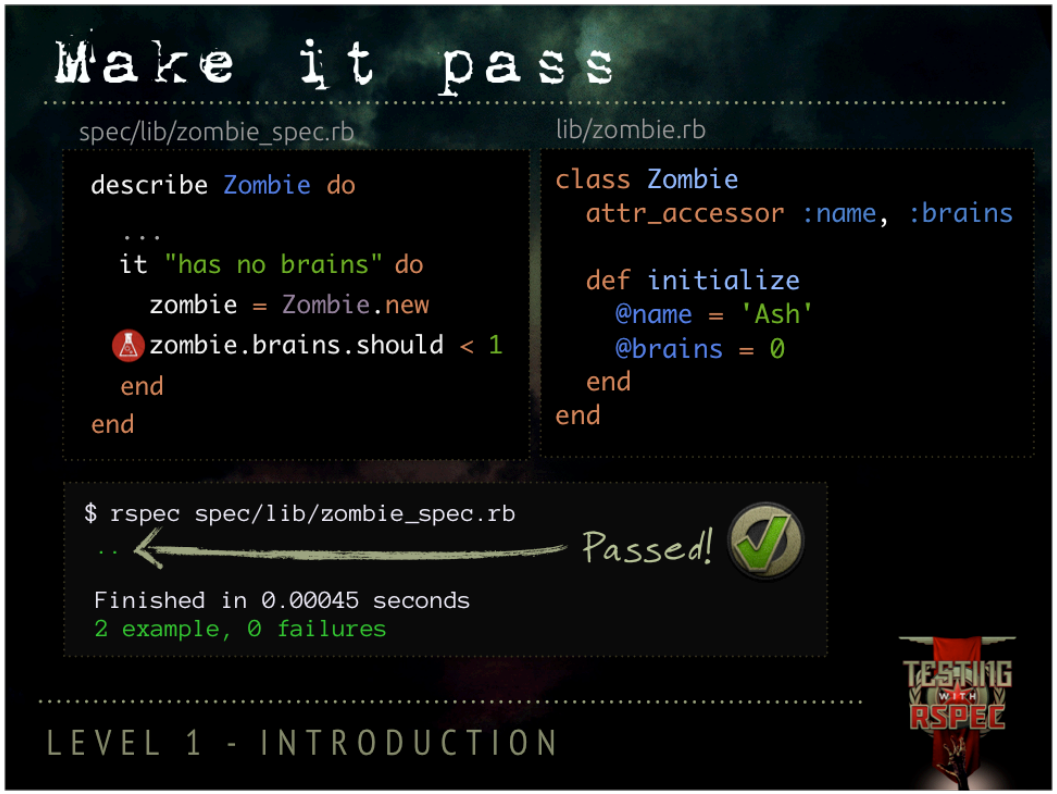


1. Add another expectation to zombie_spec.rb
2. Run rspec
3. See it fail
4. Result will say something similar to

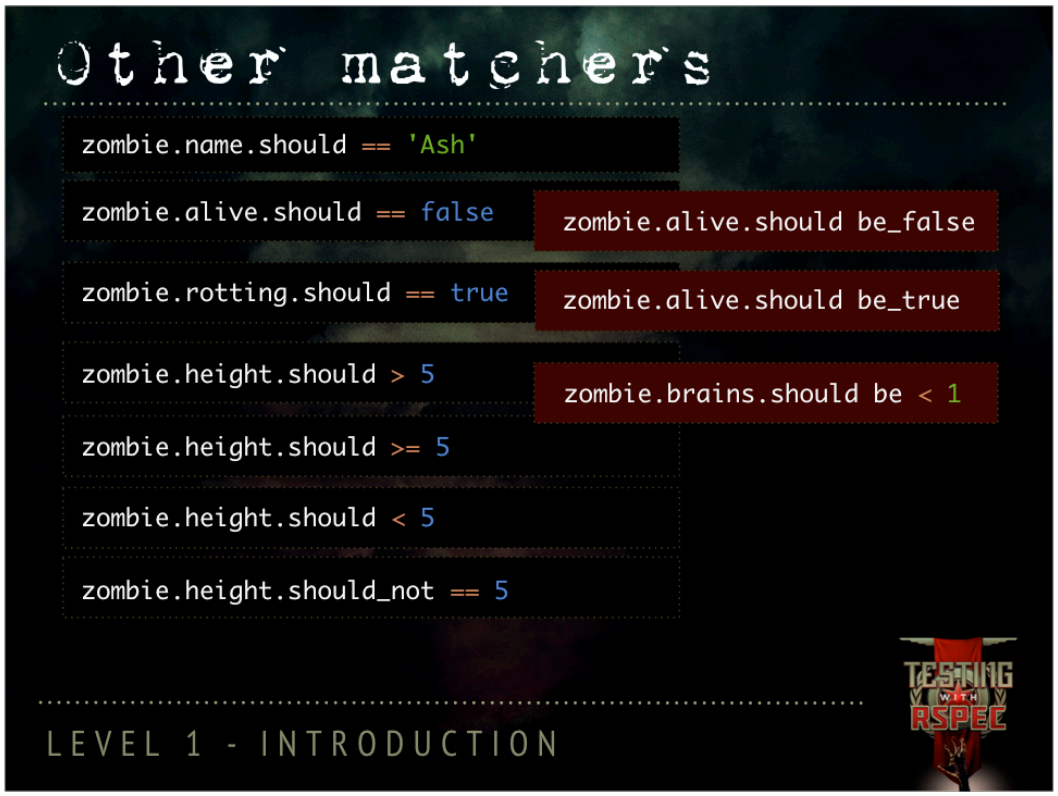
Zombie

is named Ash

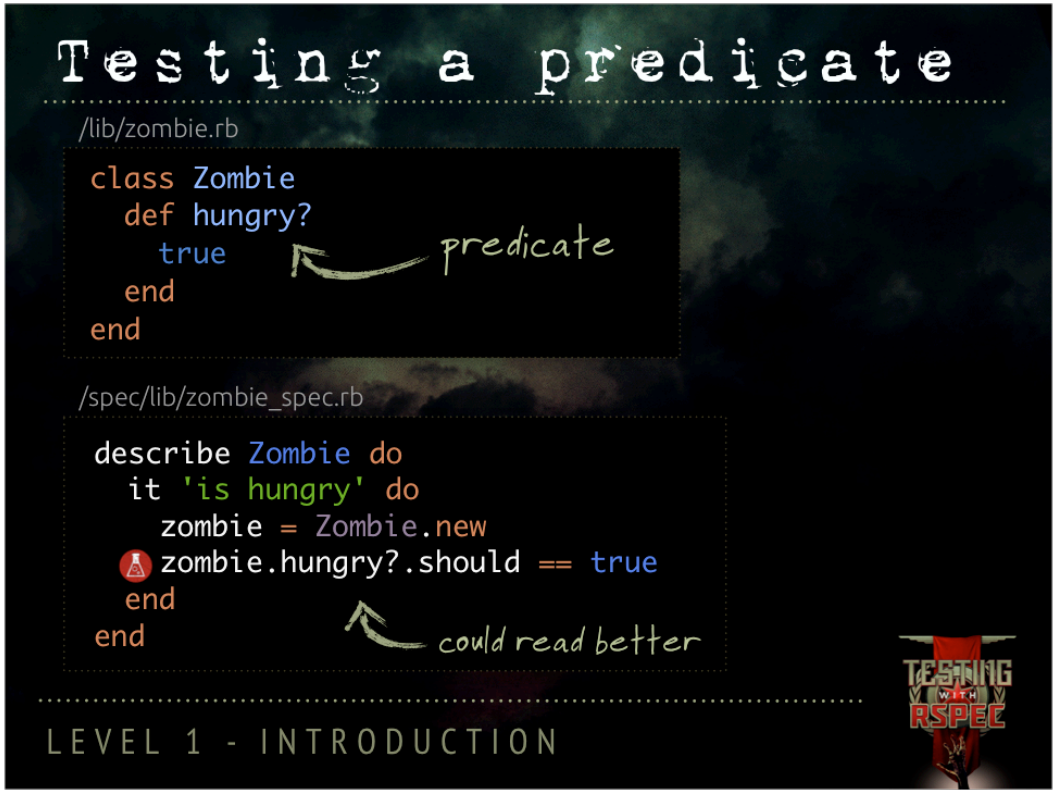
has no brains (FAILED -1)



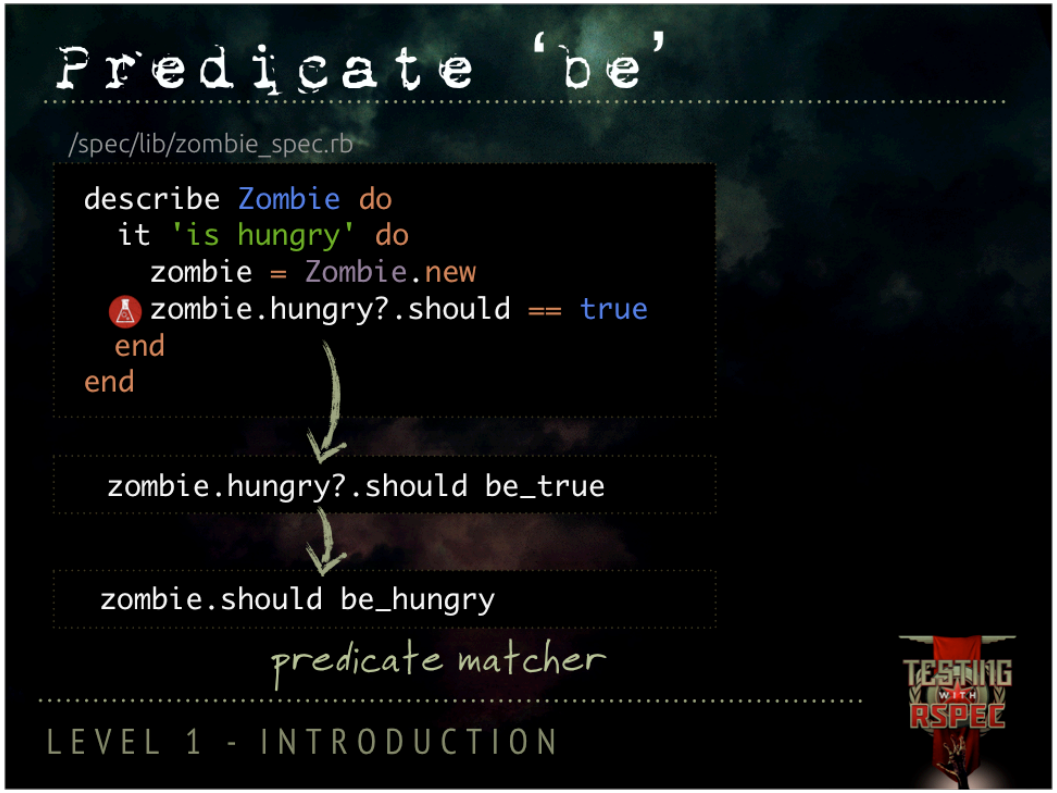
1. Add brains to zombie.rb
2. In attr_accessor
3. In initialize
4. Run rspec
5. See it pass



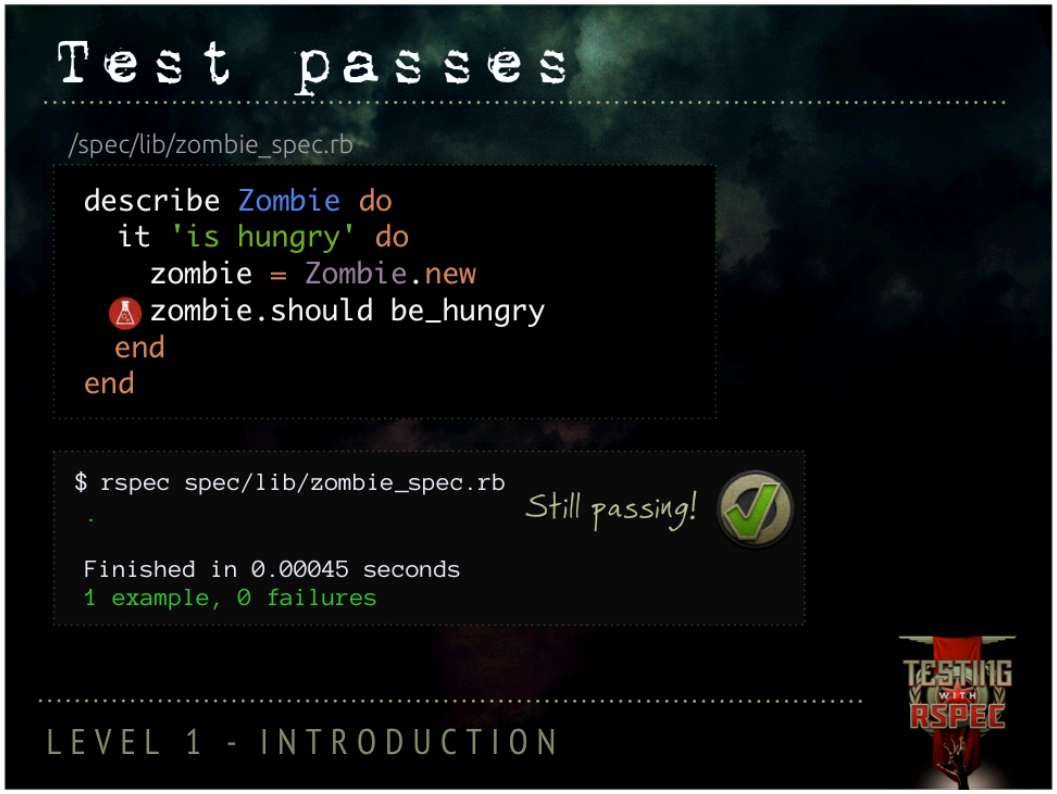
- Read through, no need to type it in



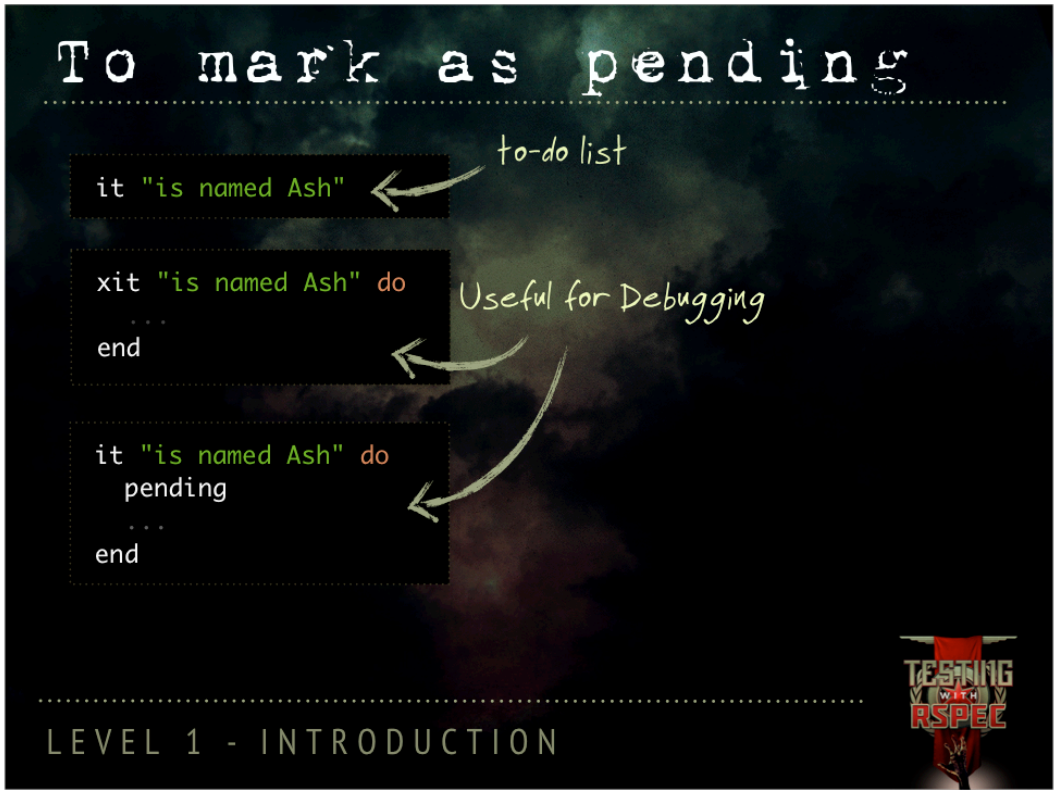
1. Add code to appropriate files
2. Run rspec
3. See it pass



1. Change appropriate line to zombie.hungry?.should be_true
2. Run rspec
3. See it pass
4. Change appropriate line to zombie.should be_hungry
5. Run rspec
6. See it pass



1. Run rspec
2. See it pass

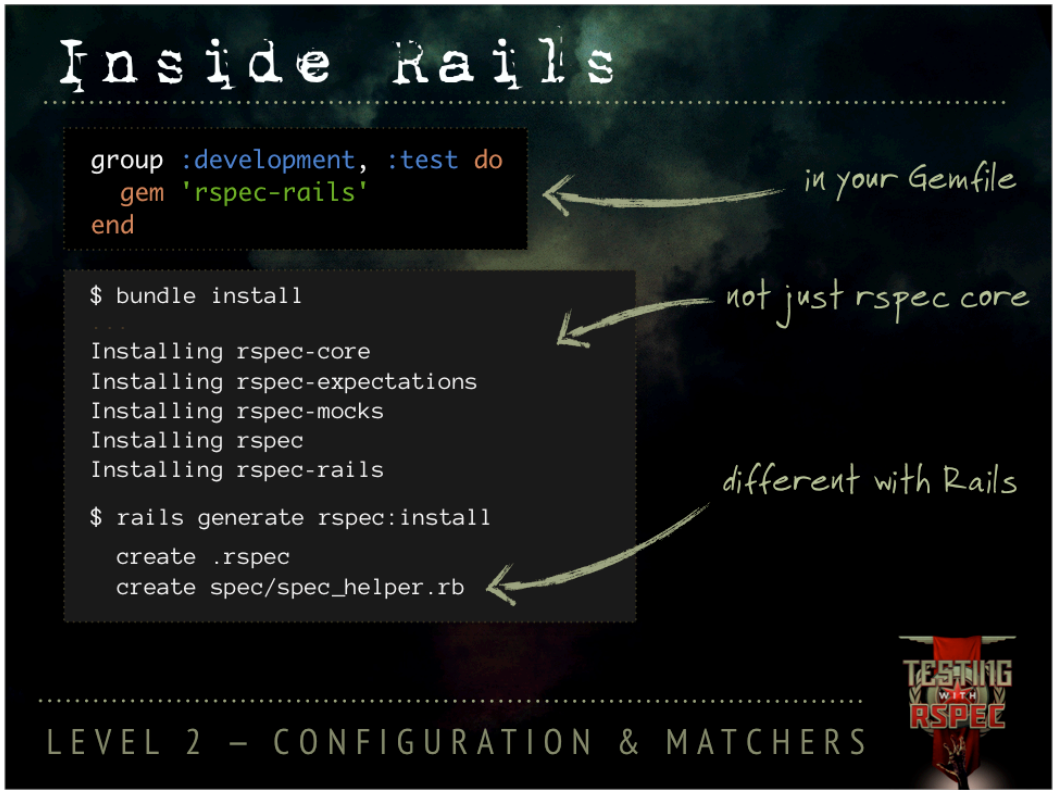


1. Create a new expectation for example:
- it “is dead”
1. Run rspec
2. See what happens
3. use xit to mark the expectation it “is named Ash” as pending
4. Run rspec
5. See what happens
6. Use keyword pending to mark it “has no brains” as pending
7. Run rspec
8. See what happens
9. Undo 3
10. Undo 6
11. Run rspec
12. Should have 4 examples, 0 failures, 1 pending
13. Submit zombie.rb (as zombie1.rb) and zombie_spec.rb (as zombie_spec1.rb) to moodle

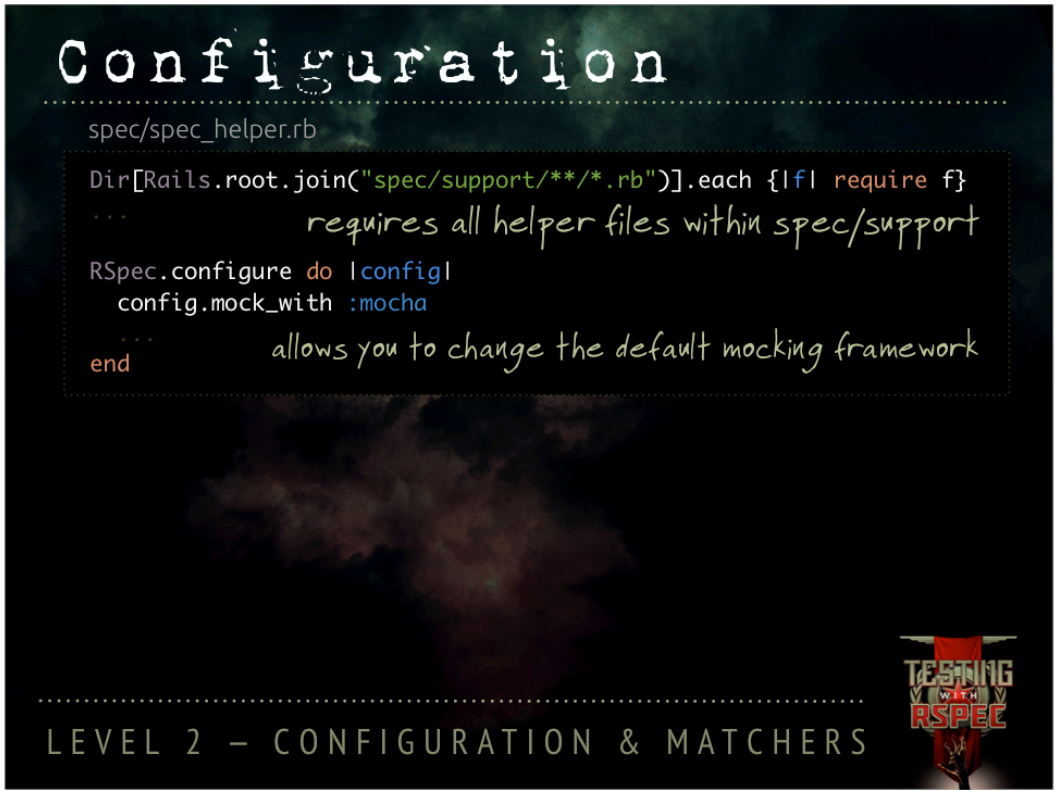
Using rspec with rails

1. Run “rails new lab4-2” in rails_project directory
2. cd lab4-2

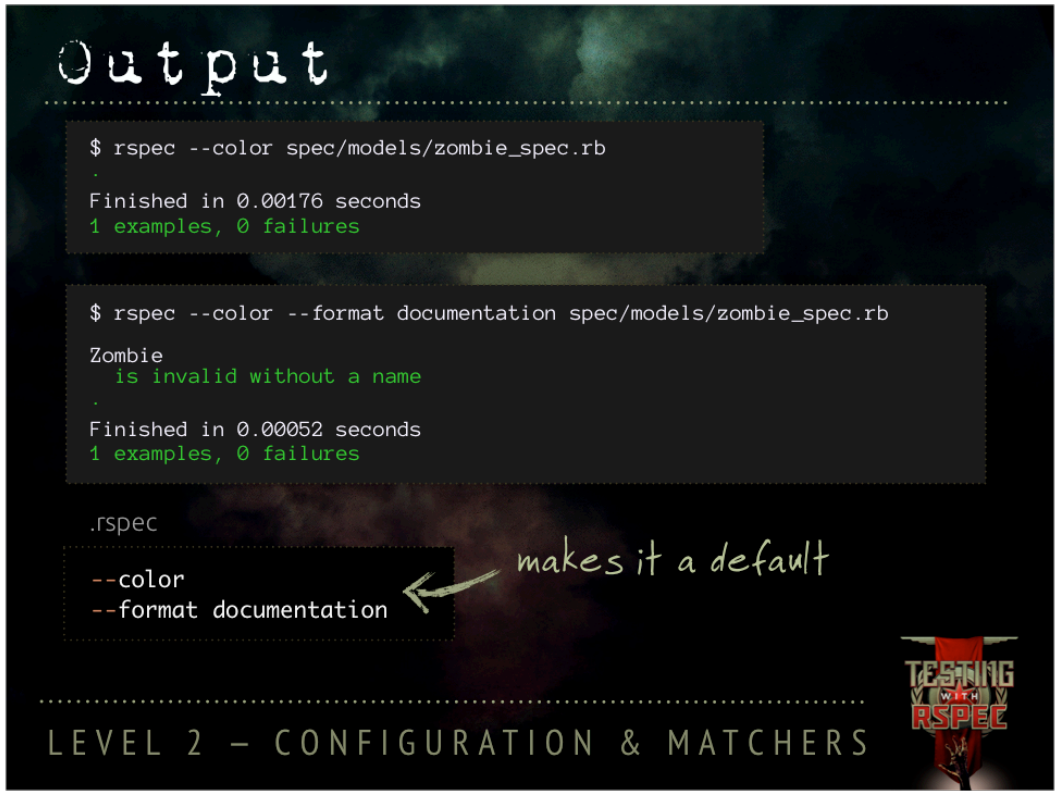
Move on to the next slide ….nothing on here



1. Add appropriate text to Gemfile
2. In group:test do at end of file add:
- gem ‘mocha’, : require => false
3. Run bundle install
4. Run rails generate rspec:install

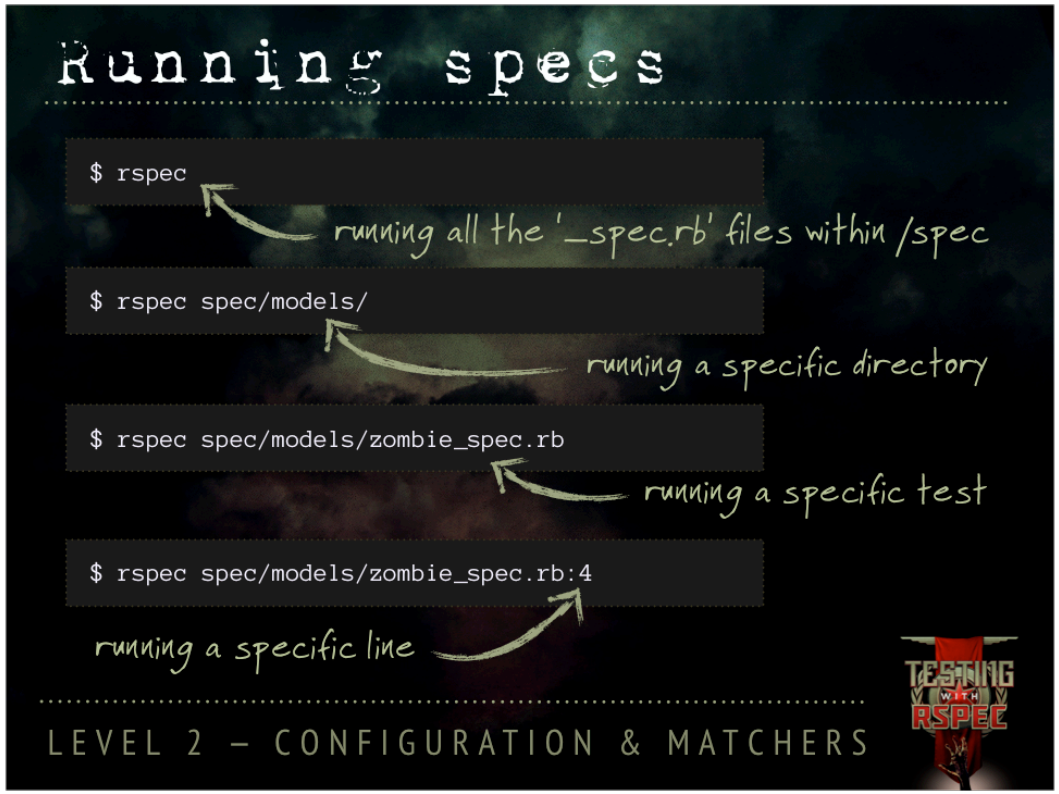


1. Open spec/spec_helper.rb
2. Uncomment config.mock_with :mocha
3. Add require 'mocha/setup' to top of spec_helper.rb …after the other require(s)
4. RUN
    1. rails generate model zombie
    2. rails generate model tweet
    3. rails generate model weapon
    4. rake db:migrate
    5. rake db:test:load



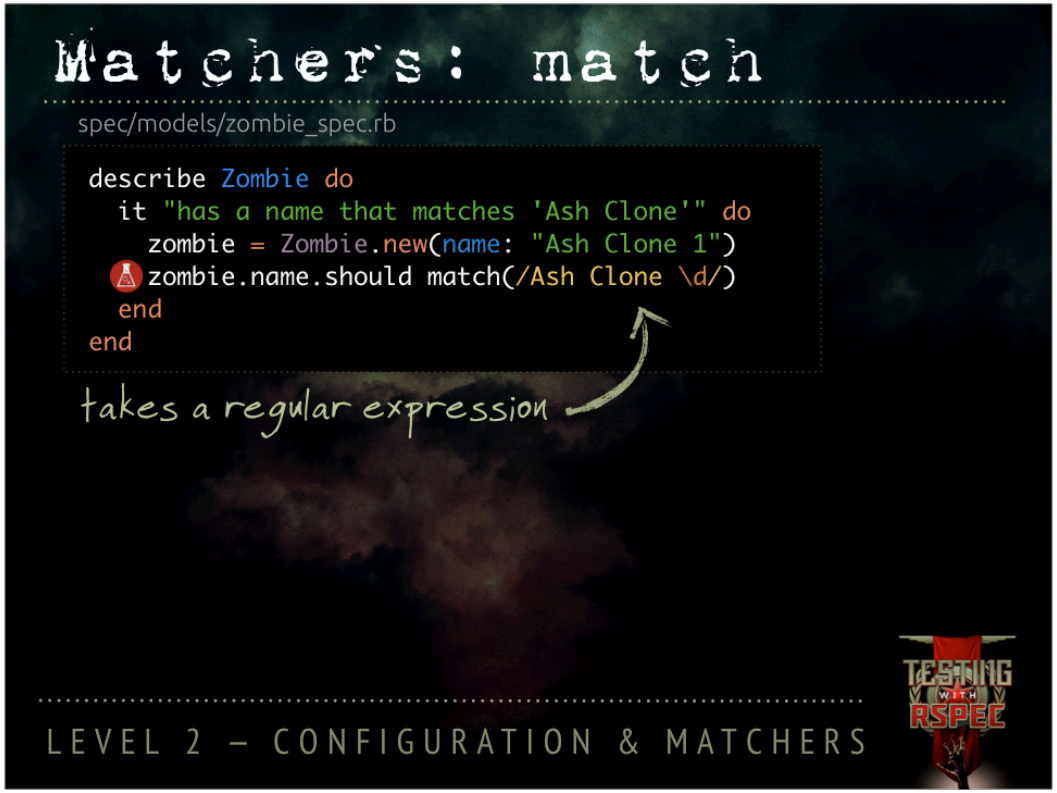
1. Set color and format for zombie_spec.rb
2. Result should say 1 pending since we haven't written any expectation or test in the file
3. For more information on format see :

https://www.relishapp.com/rspec/rspec-core/v/2-6/docs/command-line/format-option



- Simply read through, no need to run anything here

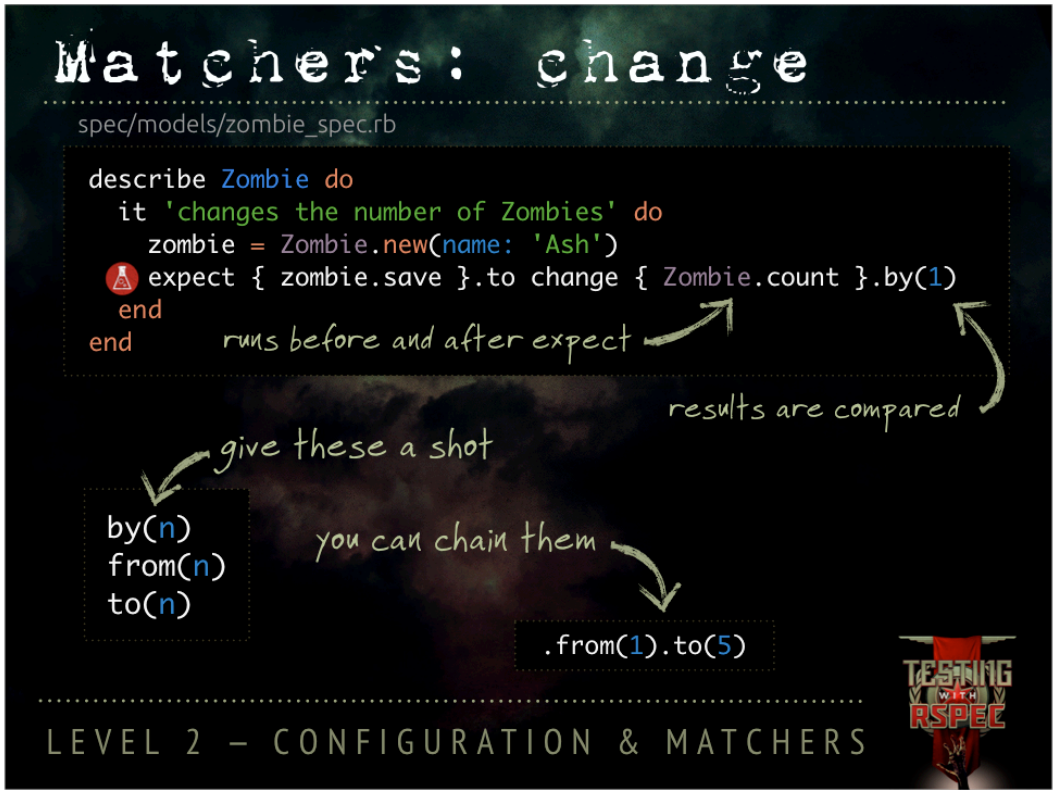
1. Add code from slide to appropriate files
2. Run rspec on zombie_spec.rb
3. See it fail
4. Add attr_accessor :name to zombie.rb
5. Run rspec on zombie_spec.rb
6. See it pass



- Add code to appropriate file
- Run rspec on zombie_spec.rb
- See it pass

1. Add code to appropriate file
2. Run rspec on zombie_spec.rb
3. See it fail
4. go into tweet.rb
5. add
6. attr_accessor :status
7. belongs_to :zombie
8. Run rspec on zombie_spec.rb
9. See it fail
10. Go into zombie.rb
11. Add :tweets after :name in attr_accessor
12. run rspec on zombie_spec.rb
13. See if pass

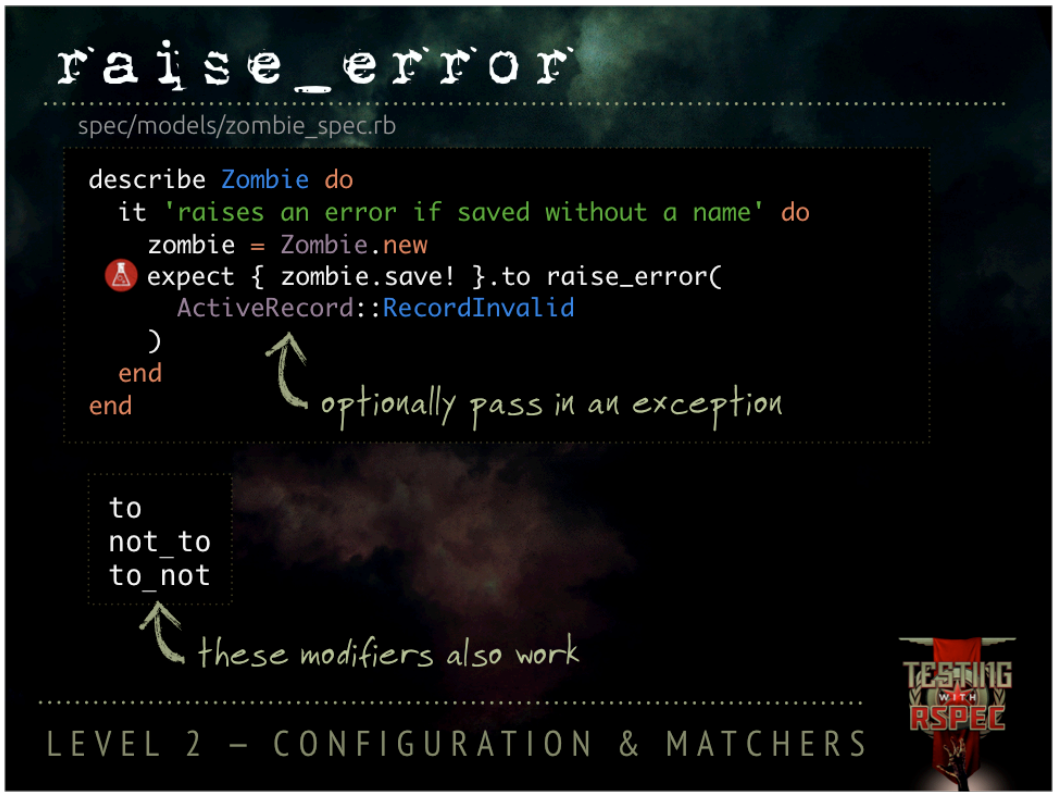
1. Go into weapon.rb and add
   attr_accessor :name
   belongs_to :zombie
2. Go into zombie.rb and add
   has_many :weapons
   def initialize(attributes = nil)
   super(attributes)
   weapons << Weapon.new
   weapons << Weapon.new
   end
3. Add *first code* from slide to appropriate file
4. Run rspec on zombie_spec.rb
5. See it fail (4 examples, 1 failure)
6. Change code to *second code" from slide
7. Run rspec on zombie_spec.rb
8. See it pass



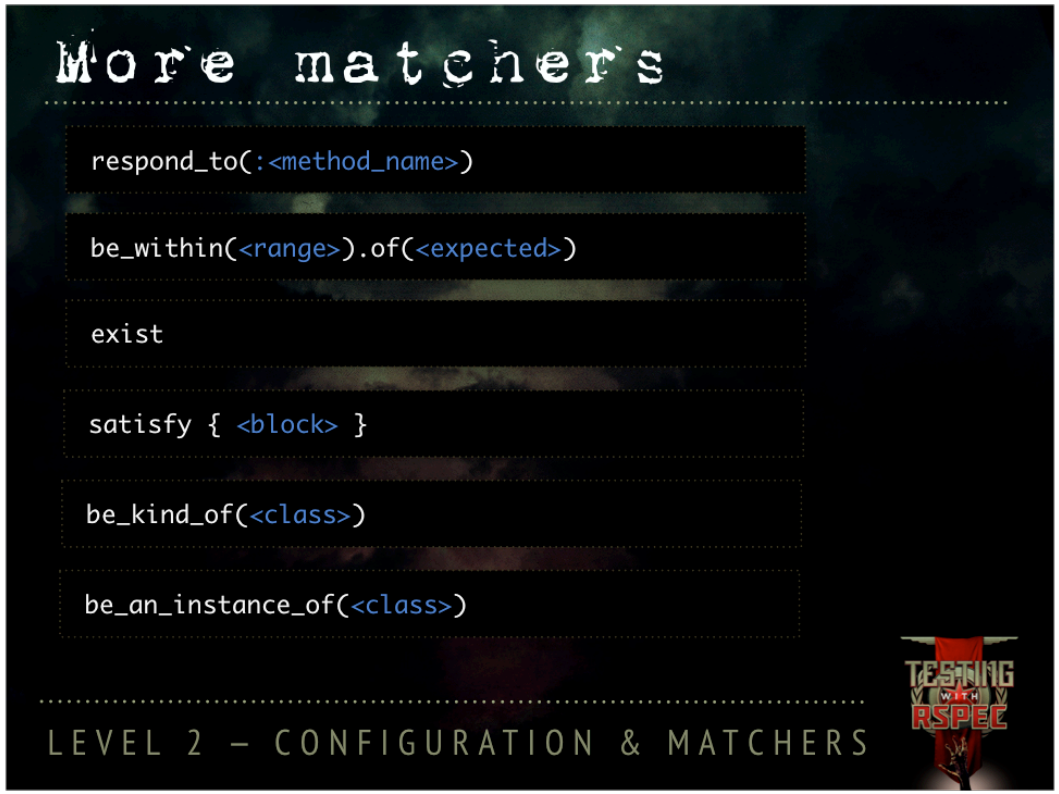
1. Add code (that uses by) to appropriate file
2. Run rspec on zombie_spec.rb
3. See it pass (5 examples, 0 failures)
4. Change code to

```
 it 'changes  the number of Zombies' do
    zombie = Zombie.new(name: "Ash")
expect { zombie.save}.to change { Zombie.count }.from(0).to(1)
  end
```

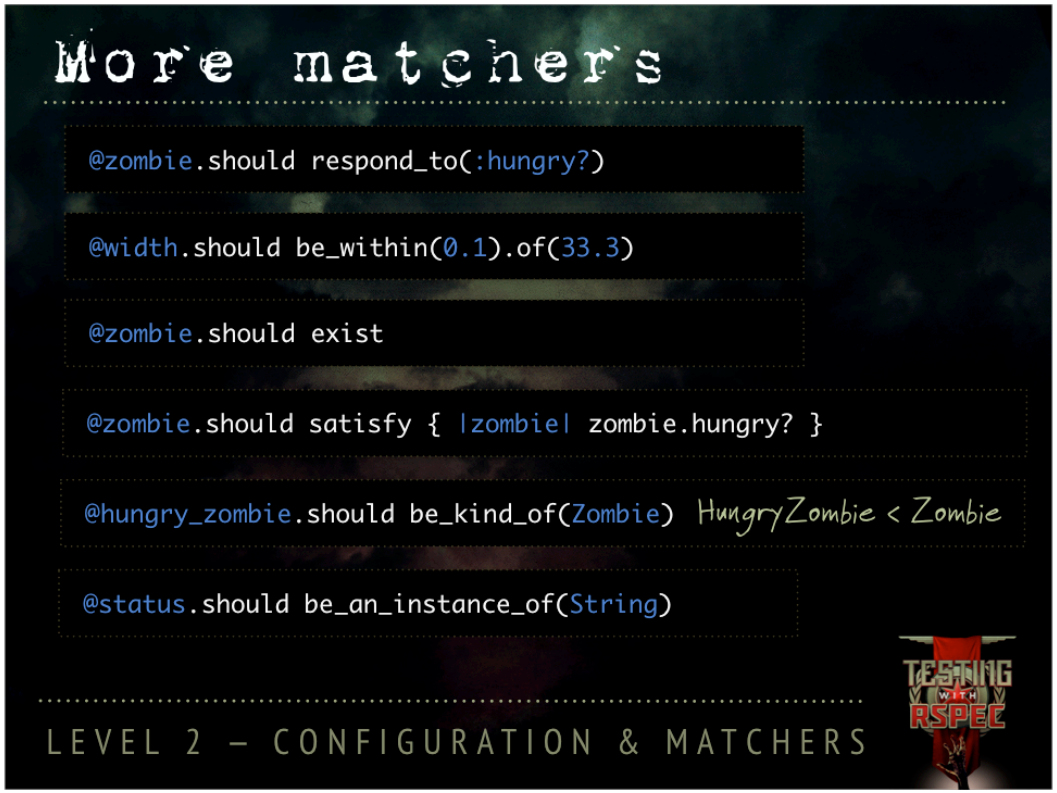
5. Run rspec on zombie_spec.rb
6. See it pass (5 examples, 0 failures



1. Add to appropriate file
2. Run rspec on zombie_spec.rb
3. See it pass (6 examples, 0 failures)



- Read through, no need to type it into a file

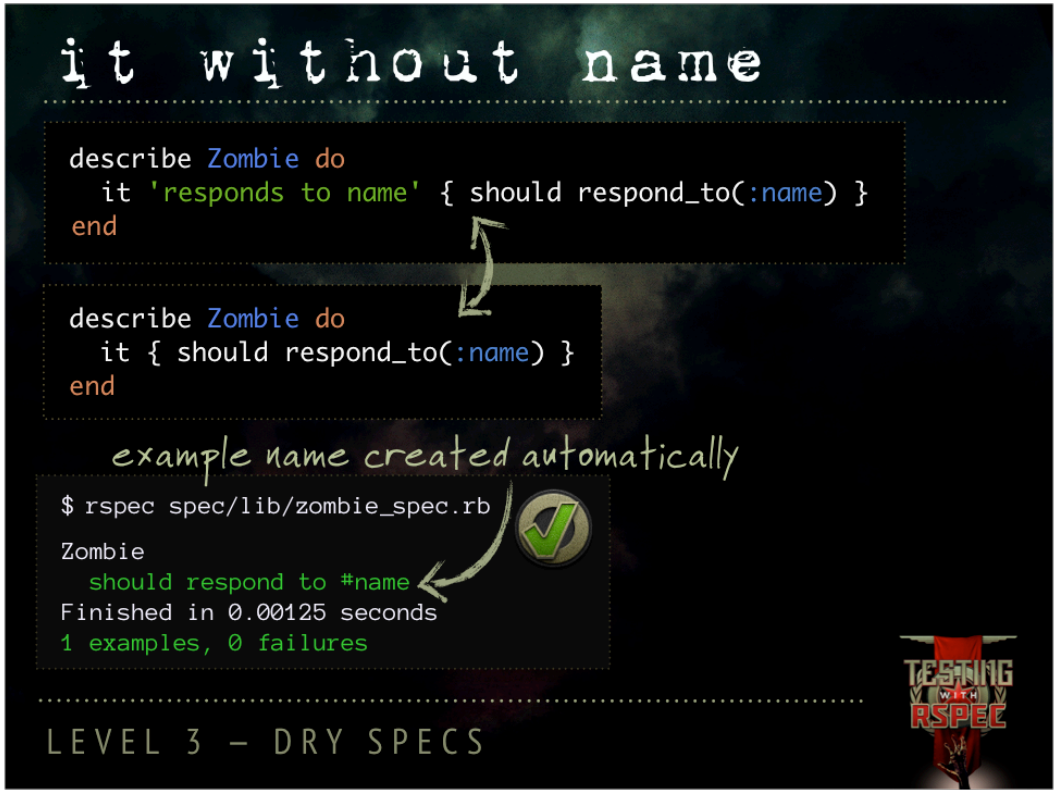


- Read through, no need to type into file

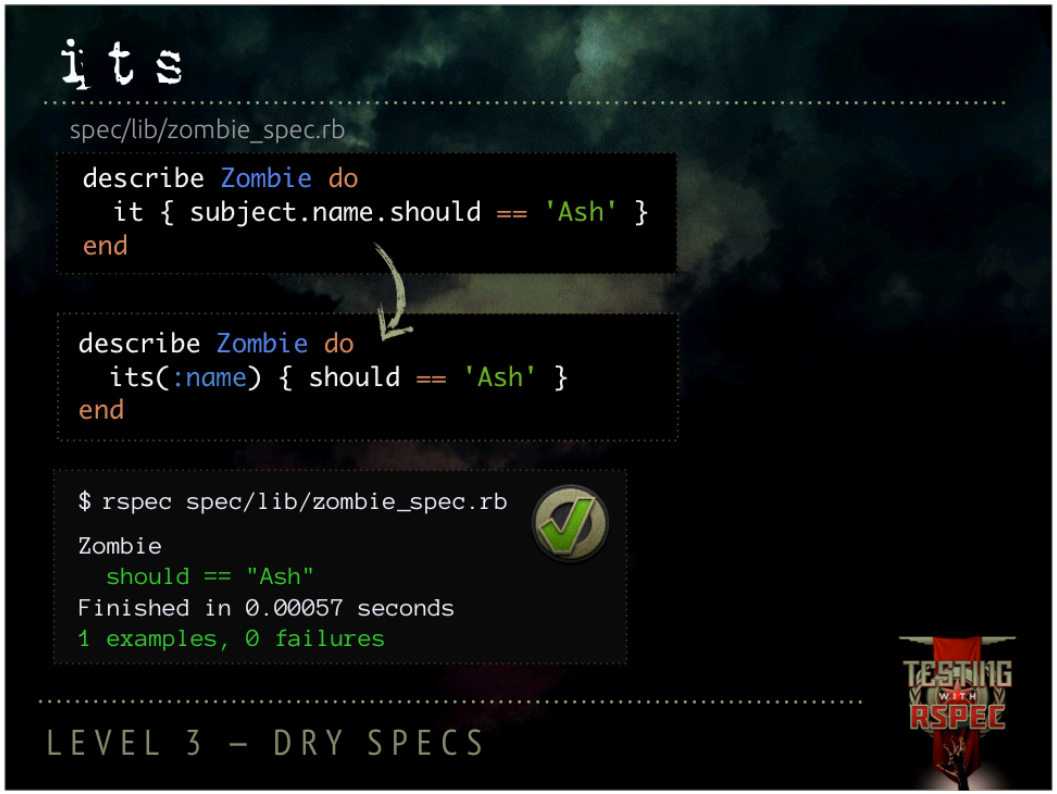

DRY Specs
• LEVEL 3 •
Don't Repeat Yourself

1. Type in file
2. Run rspec
3. See it pass (7 examples, 0 failures)
4. Change code to use subject
5. Run rspec
6. See it pass again

1. Change code to use
    1. Should respond_to(:name)
2. Run rspec
3. See it pass again (7 examples, 0 failures)



1. Change code to use
   it {should respond_to(:name)}
2. Run rspec
3. See it pass again
4. Submit zombie.rb (as zombie2.rb) and zombie_spec.rb (as zombie_spec2.rb) on moodle



Example of using its

1. Delete all the expectations from zombie_spec.rb
2. type in code from slide
3. Change initialize in zombie.rb to:

```
def initialize(attributes = nil)
  super(attributes)
  @name = 'Ash'
end
```

4. The only expectation in zombie_spec.rb should be:

```
its(:name) { should == 'Ash' }
```

5. Run rspec on zombie_spec.rb
6. See it pass

1. Change your zombie_spec.rb to be:

```
require 'spec_helper'

describe Zombie do
  its(:name) { should == 'Ash' }
  its(:brains) { should be_nil }
  its('tweets.size') { should == 2 }
end
```

2. Make all 3 examples to pass (right now only 1 is passing … the first one which we worked on in the previous slide)

3. Submit to moodle (as zombie3.rb) your updated zombie.rb that makes all three expectations pass