TESTING WITH RSPEC

Custom Matchers

• LEVEL 6 •

Refactoring

app/models/zombie.rb

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

we might do this a lot!

spec/models/zombie_spec.rb

```ruby
describe Zombie do
  it 'validates presence of name' do
    zombie = Zombie.new(name: nil)
    zombie.valid?
    zombie.errors.should have_key(:name)
  end
end
```

like 'include', but for hashes

Refactoring

app/models/zombie.rb

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

spec/models/zombie_spec.rb

```ruby
describe Zombie do
  it 'validates presence of name' do
    zombie = Zombie.new(name: nil)
    zombie.should validate_presence_of_name
  end
end
```

so let's create this custom matcher

Custom matchers

spec/support/validate_presence_of_name.rb

```
module ValidatePresenceOfName
  class Matcher
    def matches?(model)
      model.valid?
      model.errors.has_key?(:name)
    end
  end
  def validate_presence_of_name
    Matcher.new
  end
end

RSpec.configure do |config|
  config.include ValidatePresenceOfName, type: :model
end
```

same as our example

matcher available only to model specs

Custom matchers

spec/models/zombie_spec.rb

```
describe Zombie do
  it 'validates presence of name' do
    zombie = Zombie.new(name: nil)
    zombie.should validate_presence_of_name
  end
end
```

```
$ rspec spec/models/zombie_spec.rb
Zombie
  validates presence of name

Finished in 0.02884 seconds
1 example, 0 failures
```

good, but we can't use this matcher for anything else, yet...

LEVEL 6 – CUSTOM MATCHERS

Modular matchers

spec/models/zombie_spec.rb

```ruby
describe Zombie do
  it 'validates presence of name' do
    zombie = Zombie.new(name: nil)
    zombie.should validate_presence_of(:name)
  end
end
```

it doesn't accept a parameter

```
$ rspec spec/models/zombie_spec.rb
Zombie
  validates presence of name

Failures:
 1) Zombie has errors on name
  Failure/Error: zombie.should validate_presence_of(:name)
  NoMethodError:
   undefined method `validate_presence_of' for #<RSpec::Core>
   # ./spec/models/zombie.rb:5:in `block (2 levels) in <top (required)>'

1 example, 1 failure
```

Modular matchers

spec/support/validate_presence_of_name.rb

```
module ValidatePresenceOfName
  class Matcher
    def matches?(model)
      model.valid?
      model.errors.has_key?(:name)
    end
  end

  def validate_presence_of_name
    Matcher.new
  end
end
```

time to make this reusable!

Modular matchers

spec/support/validate_presence_of.rb

```ruby
module ValidatePresenceOf
  class Matcher
    def initialize(attribute)
      @attribute = attribute
    end

    def matches?(model)
      model.valid?
      model.errors.has_key?(@attribute)
    end
  end

  def validate_presence_of(attribute)
    Matcher.new(attribute)
  end
end
```

2) store the attribute

3) use your attribute

1) accept an attribute

now let's see if it works as we expect...

Modular matchers

spec/models/zombie_spec.rb

```ruby
describe Zombie do
  it 'validates presence of name' do
    zombie = Zombie.new(name: nil)
    zombie.should validate_presence_of(:name)
  end
end
```

```
$ rspec spec/models/zombie_spec.rb
Zombie
  validates presence of name

Finished in 0.02884 seconds
1 example, 0 failures
```

it does, we're green!

Matcher failures

spec/models/zombie_spec.rb

```ruby
describe Zombie do
  it 'validates presence of name' do
    zombie = Zombie.new(name: nil)
    zombie.should validate_presence_of(:name)
  end
end
```

But what happens if the validation fails?

app/models/zombie.rb

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

Matcher failures

spec/models/zombie_spec.rb

```
describe Zombie do
  it 'validates presence of name' do
    zombie = Zombie.new(name: nil)
    zombie.should validate_presence_of(:name)
  end
end
```

```
$ rspec spec/models/zombie_spec.rb
Zombie
  validates presence of name

Failures:

 1) Zombie validates presence of name
  Failure/Error: zombie.should validate_presence_of(:name)
   NoMethodError:  undefined method 'failure_message'

Finished in 0.02884 seconds
1 example, 1 failures
```

asked your matcher for a failure message

we haven't defined one yet!

Matcher failures

spec/support/validate_presence_of.rb

```ruby
module ValidatePresenceOf
  class Matcher
    def matches?(model)
      @model = model
      @model.valid?
      @model.errors.has_key?(@attribute)
    end

    def failure_message
      "#{@model.class} failed to validate :#{@attribute} presence."
    end

    def negative_failure_message
      "#{@model.class} validated :#{@attribute} presence."
    end
  end
end
```

saved for use in messages

Matcher failures

```
$ rspec spec/models/zombie_spec.rb
Zombie
  validates presence of name

Failures:

 1) Zombie validates presence of name
  Failure/Error: zombie.should validate_presence_of(:name)
    Expected Zombie to error on :name presence.

Finished in 0.02884 seconds
1 example, 1 failures
```

message from your matcher

Chained methods

app/models/zombie.rb

```ruby
class Zombie < ActiveRecord::Base
  validates :name, presence: { message: 'been eaten' }
end
```

spec/models/zombie_spec.rb

```ruby
describe Zombie do
  it 'validates presence of name' do
    zombie = Zombie.new(name: nil)
    zombie.should validate_presence_of(:name).
      with_message('been eaten')
  end
end
```

this error message should be returned on failure

Chained methods

spec/support/validate_presence_of.rb

```
class Matcher
  def initialize(attribute)
    @attribute = attribute
    @message = "can't be blank"
  end

  def matches?(model)
    @model = model
    @model.valid?
    errors = @model.errors[@attribute]
    errors.any? { |error| error == @message }
  end

  def with_message(message)
    @message = message
    self
  end
end
```

default failure message

collect errors and find a match

override failure message & return self

Chained methods

spec/models/zombie_spec.rb

```
describe Zombie do
  it 'validates presence of name' do
    zombie = Zombie.new(name: nil)
    zombie.should validate_presence_of(:name).
      with_message('been eaten')
  end
end
```

```
$ rspec spec/models/zombie_spec.rb
Zombie
  validates presence of name

Finished in 0.02884 seconds
1 example, 0 failures
```


Chained methods

```
it { should validate_presence_of(:name).with_message('oh noes') }
it { should ensure_inclusion_of(:age).in_range(18..25) }
it { should have_many(:weapons).dependent(:destroy) }
it { should have_many(:weapons).class_name(OneHandedWeapon) }
```

single line chaining

```
it 'has many Tweets' do
  should have_many(:tweets).
    dependent(:destroy).
    class_name(Tweet)
end
```

multiple line chaining