automated testing with RSpec

RSpec

- A popular ruby testing framework
- Less Ruby-like, natural syntax
- Well formatted output

describe block

Source code at lib/user.rb

Specifications at spec/lib/user_spec.rb

spec/lib/user_spec.rb

```
describe "A User" do

end
```

describe it

spec/lib/user_spec.rb

```
describe "A User" do
  it "is named John"
  end
```

pending

spec/lib/user_spec.rb

```
describe "A User" do
  it "is named John"
end
```

```
$ rspec spec/lib/user_spec.rb
*
Pending:
      A User is named John
   # Not yet implemented
   # ./spec/lib/user_spec.rb:4
Finished in 0.00026 seconds
1 example, 0 failures, 1 pending
```

describe class

spec/lib/user_spec.rb

```
describe User do
  it 'is named John'
end
```

```
$ rspec spec/lib/user_spec.rb
user_spec.rb:3:in `<top (required)>': uninitialized constant User
(NameError)
```

create class

spec/lib/user_spec.rb

```ruby
require 'spec_helper'
describe User do
  it 'is named John'
end
```

lib/user.rb

```ruby
class User
  #empty class
end
```

```
$ rspec spec/lib/user_spec.rb
*
Pending:
      User is named John
   # Not yet implemented
   # ./spec/lib/user_spec.rb:5
Finished in 0.00059 seconds
1 example, 0 failures, 1 pending
```

expectations

spec/lib/user_spec.rb

```
require 'user'
describe User do

  it 'is named John' do
     user = User.new
      expect(user.name).to eq ‘John’
  end
end
```

```
$ rspec spec/lib/user_spec.rb
F
Failures:
 1) User is named John
   Failure/Error: user.name.should == "  John“
   NoMethodError: undefined method `name' for #<User:0x0000000129e540>…
```

success!

spec/lib/user_spec.rb

```
require 'user'
describe User do
   it 'is named John' do
       user = User.new
       Expect(user.name).to eq "John"
   end
end
```

lib/user.rb

```
class User
  attr_accessor :name

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

```
$ rspec spec/lib/user_spec.rb
.
Finished in 0.00074 seconds
1 example, 0 failures
```

expectation (one more!)

spec/lib/user_spec.rb

```
require 'user'
describe User do
   ...
  it 'has 2 elbows' do
    user = User.new
    expect(user.elbows).to
      be > 0
     expect(user.elbows).to
      eq 2

  end
end
```

```
$ rspec spec/lib/user_spec.rb
..
Finished in 0.00049 seconds
2 examples, 0 failures
```

lib/user.rb

```
   class User
    attr_accessor :name,
  :hands

  def initialize
      @name = 'John'
      @elbows = 1
    end
  end
```

more matchers

```
expect(user.name).to == "John"

expect("a string").to include("a")

expect(user.cleared_for_travel?).to be_true

expect(user).to be_cleared_for_travel

expect([1, 3, 5]).to all( be_odd )

expect(area_of_circle).to be_within(0.1).of(28.3)

expect(user.age).to_not be <= 18
```

hook

spec/models/user_spec.rb

```
describe User do
  let(:user){ User.new }

  it "is active" do
    user.active!
    expect(user).to be_active
  end

  it "can login" do
    user.active!
    Expect(user).be_able_to_login
  end
end
```

spec/models/user_spec.rb

```
describe User do
  let(:user){ User.new }
  before { user.active! }

  it "is active" do
    expect(user).to be_active
  end

  it "can login" do
    Expect(user).be_able_to_login
  end
end
```

hook (Contd.)

before(:each)

before(:all)

after(:each)

after(:all)