Behavior Driven Development w/ RSpec

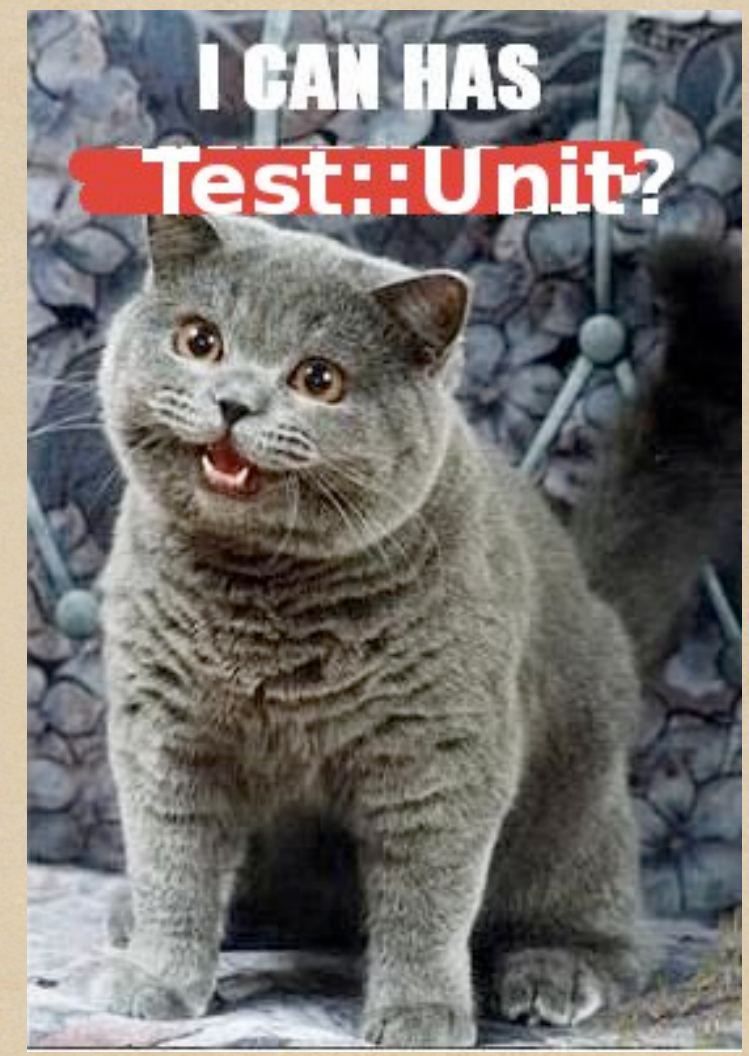
Evan David Light

evan@tiggerpalace.com

http://evan.tiggerpalace.com

What is BDD?

- More than just "writing the test first"
- It's "getting the words right"
- Focuses on business value
- Behavior comes first, testing second


I CAN HAS
Test::Unit?

Well you could but...

- Test::Unit does not help you define behavior
- RSpec does
- ... and you can still unit test in RSpec
- ... and it has mocking and stubbing support baked in

- ... and you can coverage test with RSpec (using RCov) easily
- ... and it plays nicely with Heckle
- ... and you will (later) be able to integration test in RSpec
- ... and ... and ... and ...

Holy crap!

That's a lot of stuff!

DON'T PANIC!

RSpec is easy to learn

Sample Spec

```
include Adder

describe "My adder module" do
  it "should provide a method called 'sum'" do
    Kernel.should respond_to(:sum)
  end

  it "should accept two numbers" do
    lambda { sum(1, 2) }.should_not raise_error
  end

  it "should error if given a non-number" do
    lambda { sum("foo", 1) }.should raise_error(TypeError)
    lambda { sum(1, "bar") }.should raise_error(TypeError)
    lambda { sum("foo", "bar") }.should raise_error(TypeError)
  end

  it "should return a number" do
    val = sum(1, 2)
    val.should_not be_nil
    val.should be_a_kind_of(Numeric)
  end

  it "should return the sum of its two parameters" do
    sum(1,2).should == 3
  end
end
```

Discuss!

```
include Adder

describe "My adder module" do
  it "should provide a method called 'sum'" do
    Kernel.should respond_to(:sum)
  end

  it "should accept two numbers" do
    lambda { sum(1, 2) }.should_not raise_error
  end

  it "should error if given a non-number" do
    lambda { sum("foo", 1) }.should raise_error(TypeError)
    lambda { sum(1, "bar") }.should raise_error(TypeError)
    lambda { sum("foo", "bar") }.should raise_error(TypeError)
  end

  it "should return a number" do
    val = sum(1, 2)
    val.should_not be_nil
    val.should be_a_kind_of(Numeric)
  end

  it "should return the sum of its two parameters" do
    sum(1,2).should == 3
  end
end
```

Here's how you run it

```
[evan@Timmy:sample1]$ spec adder_spec.rb
.....

Finished in 0.005582 seconds

5 examples, 0 failures
[evan@Timmy:sample1]$
```

Ooh! That was hard!

What if I want to show my customer that I'm making progress?

```
[evan@Timmy:sample1]$ spec adder_spec.rb --format specdoc

My adder module
- should provide a method called 'sum'
- should accept two numbers
- should error if given a non-number
- should return a number
- should return the sum of its two parameters

Finished in 0.00537 seconds

5 examples, 0 failures
[evan@Timmy:sample1]$
```

Let's see what happens if we were to write a placeholder description in our spec...

```
it "should do something else"
```

```
[evan@Timmy:sample1]$ spec adder_spec.rb --format specdoc

My adder module
- should provide a method called 'sum'
- should accept two numbers
- should error if given a non-number
- should return a number
- should return the sum of its two parameters
- should do something else (PENDING: Not Yet Implemented)

Finished in 0.005466 seconds

6 examples, 0 failures, 1 pending

Pending:
My adder module should do something else (Not Yet Implemented)
[evan@Timmy:sample1]$
```

WAA! I write Rails apps!

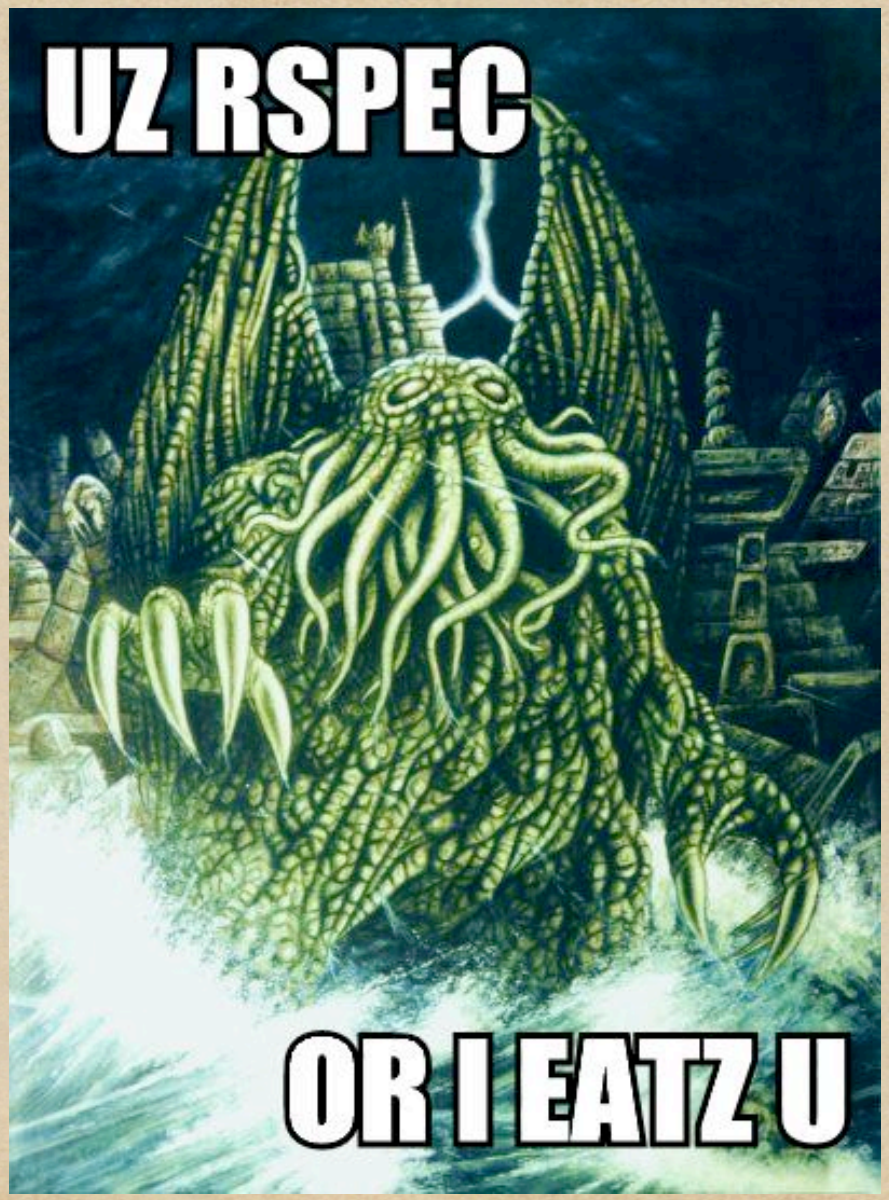
- No problem!
- RSpec and RSpec-on-Rails available as Rails plugins
- RSpec-on-Rails includes several handy-dandy generators

RSpec-on-Rails

Demo

RSpec looks awesome! But how do I define behavior at a high level?

- Story Runner
- Currently in development...
- ... but it is available now from within edge RSpec
- Caveat: the Story Runner DSL is still evolving!


UZ RSPEC
OR I EATZ U