Table of Contents (left)

RSpec Expectations

RSpec::Expectations lets you express expected outcomes on an object in an example.

```
account.balance.should eq(Money.new(37.42, :USD))
```

Install

If you want to use rspec-expectations with rspec, just install the rspec gem and RubyGems will also install rspec-expectations for you (along with rspec-core and rspec-mocks):

```
gem install rspec
```

If you want to use rspec-expectations with another tool, like Test::Unit, Minitest, or Cucumber, you can install it directly:

```
gem install rspec-expectations
```

Basic usage

Here's an example using rspec-core:

```
describe Order do
  it "sums the prices of the items in its line items" do
    order = Order.new
    order.add_entry(LineItem.new(:item => Item.new(
      :price => Money.new(1.11, :USD)
    )))
    order.add_entry(LineItem.new(:item => Item.new(
      :price => Money.new(2.22, :USD),
      :quantity => 2
    )))
    order.total.should eq(Money.new(5.55, :USD))
  end
end
```

The `describe` and `it` methods come from rspec-core. The `Order`, `LineItem`, and `Item` classes would be from *your* code. The last line of the example expresses an expected outcome. If `order.total == Money.new(5.55, :USD)`, then the example passes. If not, it fails with a message like:

```
expected: #<Money @value=5.55 @currency=:USD>
     got: #<Money @value=1.11 @currency=:USD>
```

Built-in matchers

Equivalence

```
actual.should eq(expected)  # passes if actual == expected
actual.should == expected   # passes if actual == expected
actual.should eql(expected) # passes if actual.eql?(expected)
```

Identity

```
actual.should be(expected)    # passes if actual.equal?(expected)
actual.should equal(expected) # passes if actual.equal?(expected)
```

Comparisons

```
actual.should be >  expected
actual.should be >= expected
actual.should be <= expected
actual.should be <  expected
actual.should be_within(delta).of(expected)
```

Regular expressions

```
actual.should =~ /expression/
actual.should match(/expression/)
```

Types/classes

```
actual.should be_an_instance_of(expected)
```

```
actual.should be_a_kind_of(expected)
```

Truthiness

```
actual.should be_true  # passes if actual is truthy (not nil or false)
actual.should be_false # passes if actual is falsy (nil or false)
actual.should be_nil   # passes if actual is nil
```

Expecting errors

```
expect { ... }.to raise_error
expect { ... }.to raise_error(ErrorClass)
expect { ... }.to raise_error("message")
expect { ... }.to raise_error(ErrorClass, "message")
```

Expecting throws

```
expect { ... }.to throw_symbol
expect { ... }.to throw_symbol(:symbol)
expect { ... }.to throw_symbol(:symbol, 'value')
```

Predicate matchers

```
actual.should be_xxx         # passes if actual.xxx?
actual.should have_xxx(:arg) # passes if actual.has_xxx?(:arg)
```

See RSpec::Matchers for more about predicate matchers.

Ranges (Ruby >= 1.9 only)

```
(1..10).should cover(3)
```

Collection membership

```
actual.should include(expected)
```

Examples

```
[1,2,3].should include(1)
[1,2,3].should include(1, 2)
{:a => 'b'}.should include(:a => 'b')
"this string".should include("is str")
```

Learn more

See RSpec::Expectations for more information about `should` and `should_not` and how they work.

See RSpec::Matchers for more information about the built-in matchers that ship with rspec-expectations, and how to write your own custom matchers.

Also see

- http://github.com/rspec/rspec

- http://github.com/rspec/rspec-core
- http://github.com/rspec/rspec-mocks

Generated on Thu Jan 5 00:11:52 2012 by yard 0.7.4 (ruby-1.9.3).