The Pragmatic Studio

Test-Driven Development with Rails

Warranty:

Sample code is provided for educational purposes only, and comes with absolutely no warranty. It should not be used in production applications.

Kick Start

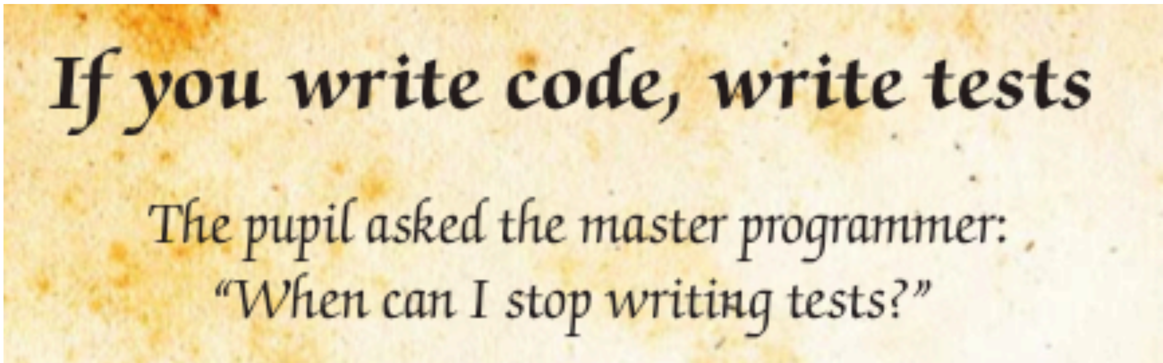
Who We Are

EdgeCase
software artisans

Joe O'Brien is a father, speaker, author and developer. Before helping found EdgeCase, LLC, Joe was a developer with ThoughtWorks and spent much of his time working with large J2EE and .NET systems for Fortune 500 companies.

Jim Weirich has been active in the software development world for over twenty-five years, with experience that ranges from real-time data acquisition for jet engine testing to image processing and web services for the financial industry. He is currently the chief scientist for EdgeCase, LLC.

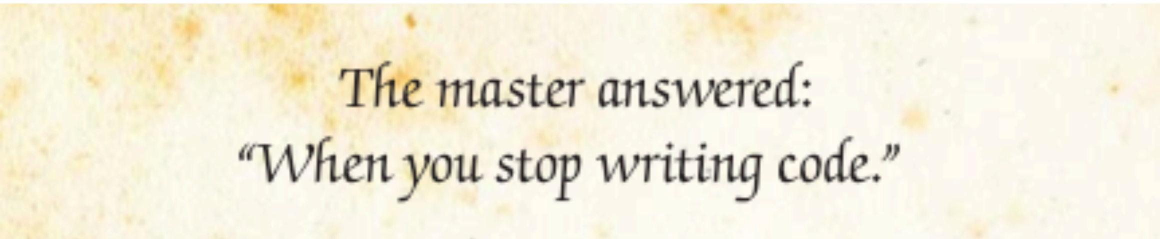
The Way of Testivus



If you write code, write tests

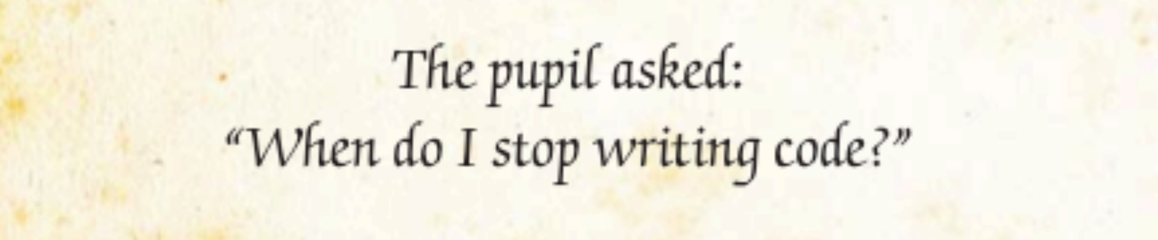
The pupil asked the master programmer:
"When can I stop writing tests?"

The Way of Testivus



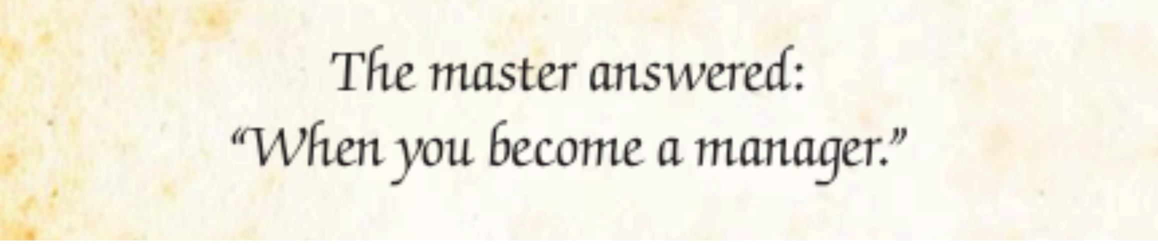
The master answered:
"When you stop writing code."

The Way of Testivus



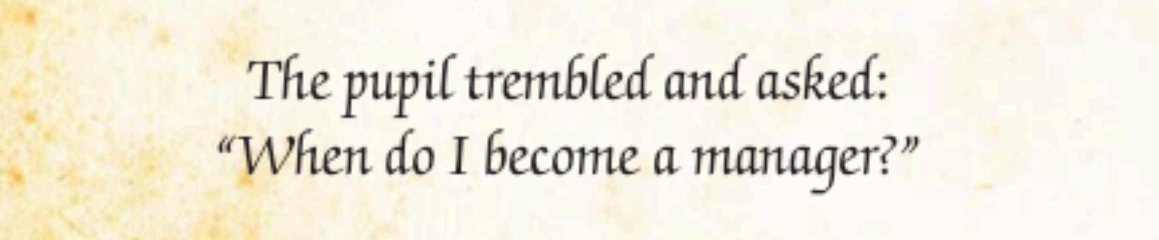
The pupil asked:
"When do I stop writing code?"

The Way of Testivus



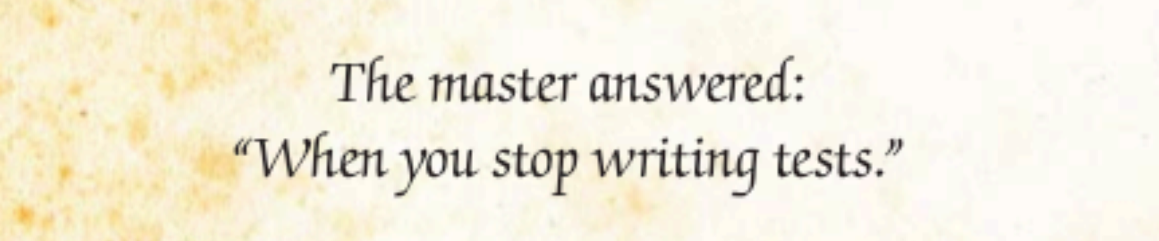
The master answered:
"When you become a manager."

The Way of Testivus



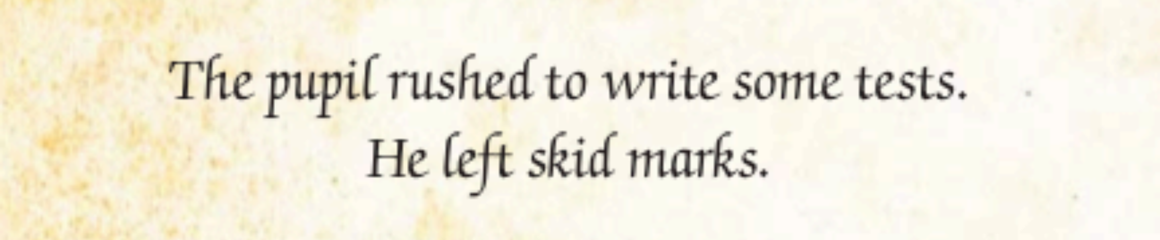
The pupil trembled and asked:
"When do I become a manager?"

The Way of Testivus



The master answered:
"When you stop writing tests."

The Way of Testivus



The pupil rushed to write some tests.
He left skid marks.

Testimonial

- More Confidence
- Better APIs
  - Tests guide design
- Less Debugging

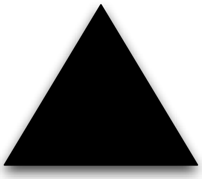
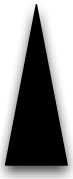
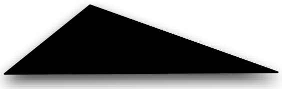
Testing Becomes Fun

LAB: Triangle

- Function: classify_triangle(a, b, c)
- The arguments a, b, and c are numbers representing the lengths of the three sides of a triangle
- The function should classify the triangle by examining the lengths of the sides and return:
    - :equilateral, :isosceles, or :scalene

LAB: Triangle

- Function: classify_triangle(a, b, c)

:equilateral		3 equal sides
:isosceles		2 equal sides
:scalene		all sides different

LAB 1: Triangle

Make a list of test cases to test the function

a	b	c	expected
3	3	3	:equilateral

Triangle History

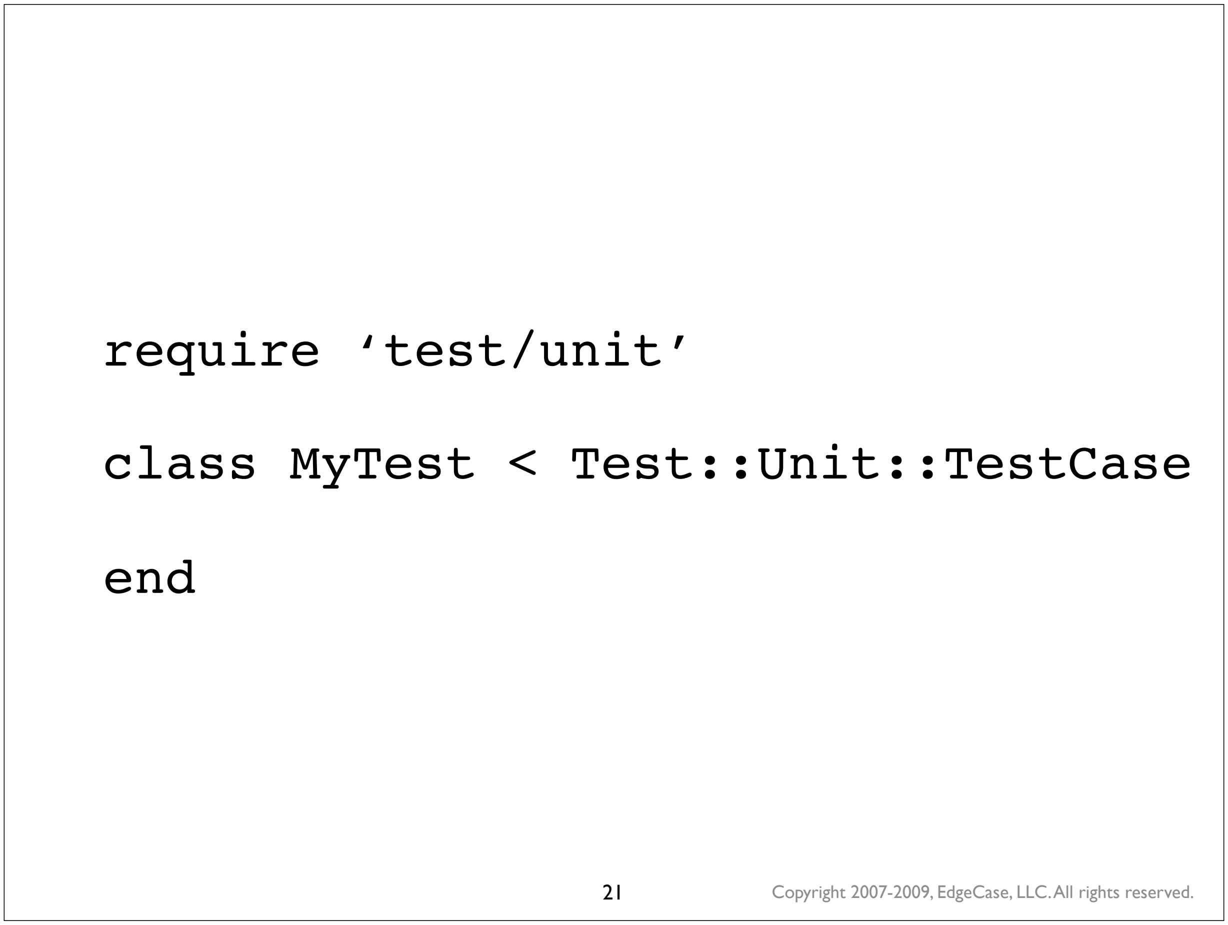
- Glenford Myers (The Art of Software Testing): 14 Tests
- Bob Binder (Testing Object Oriented Software Systems): 65 Tests
- Kent Beck: 6 Tests
- JUF: 25 Tests

What Kind of Tests did You Write?

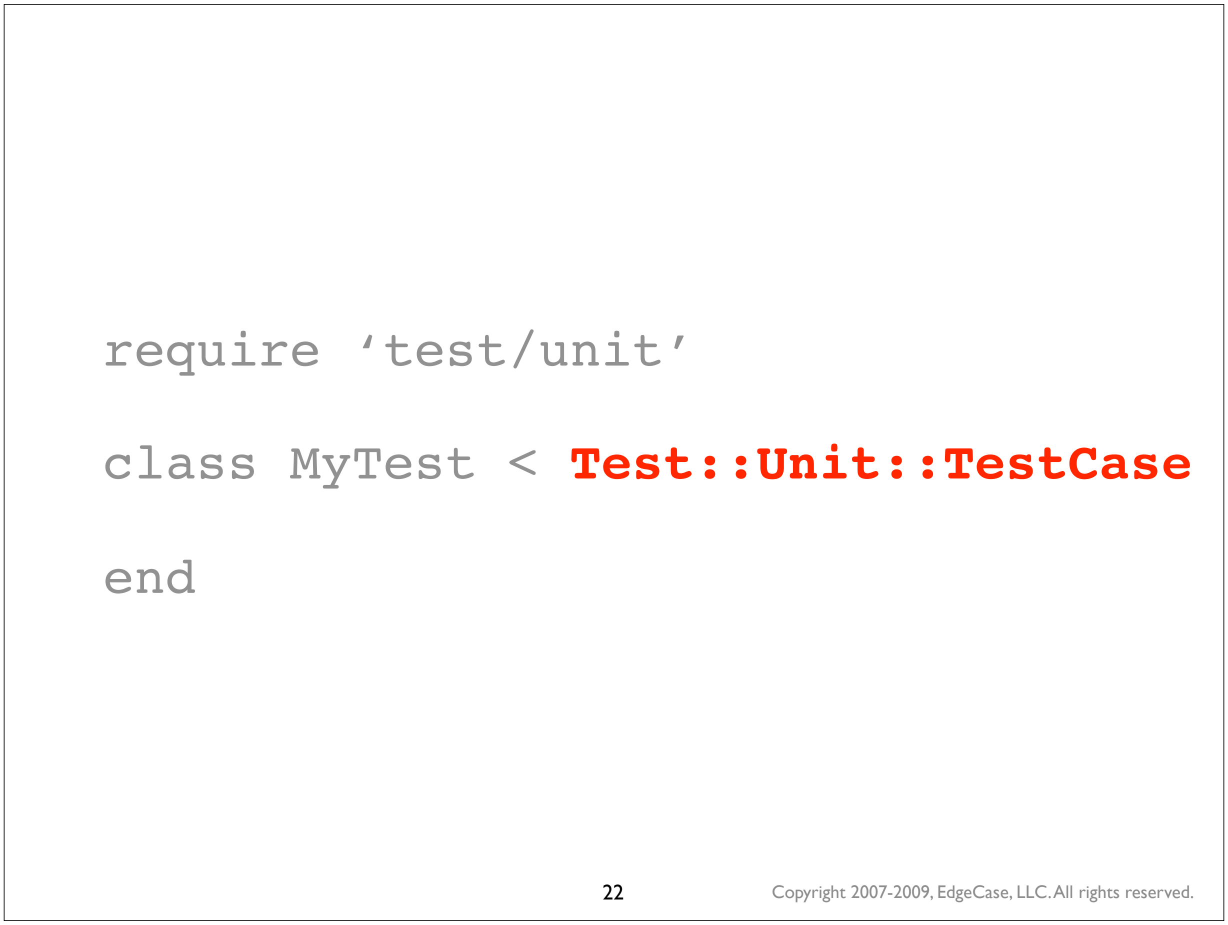
test/unit

Ruby’s xUnit Framework

```
require ‘test/unit’
```



```
require ‘test/unit’

class MyTest < Test::Unit::TestCase

end
```



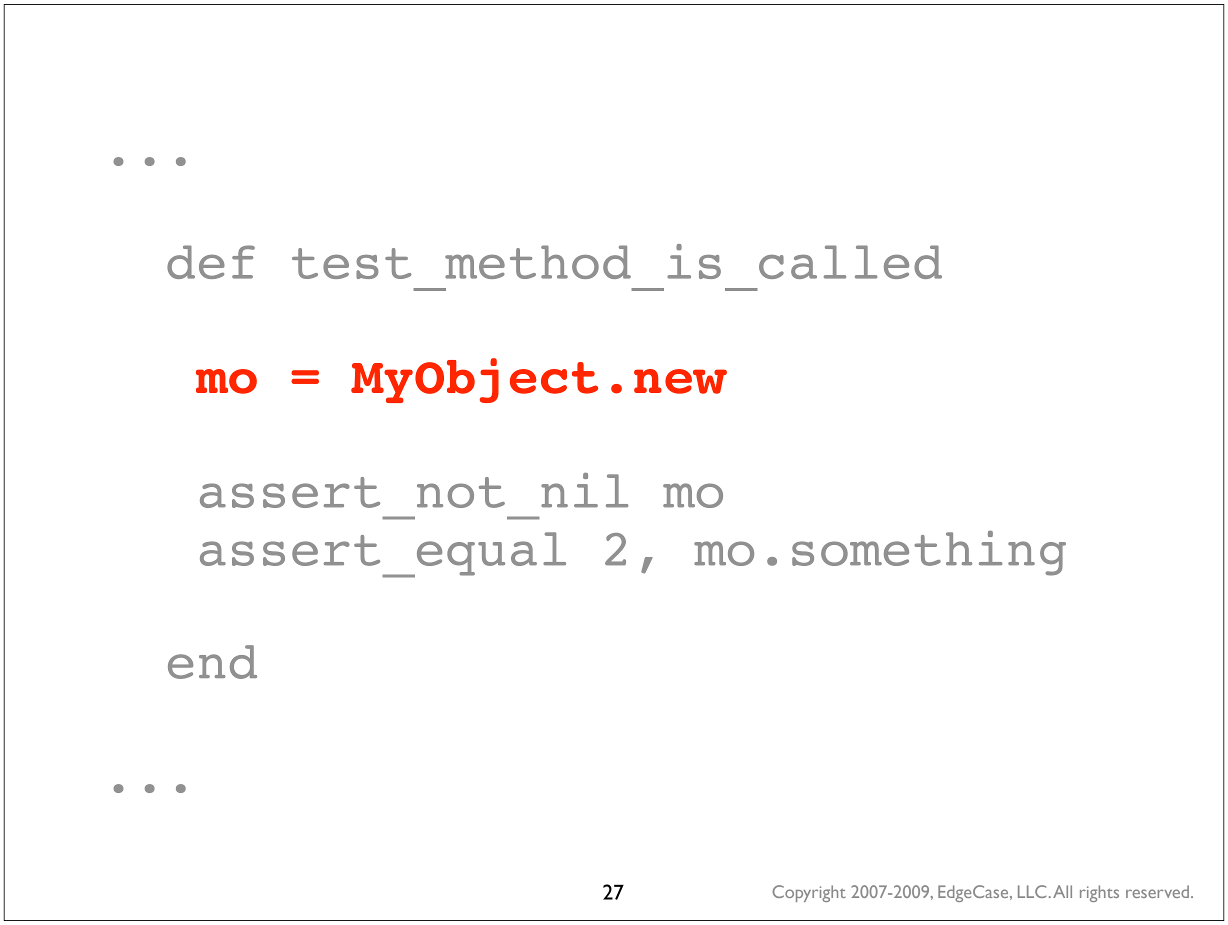
```
require ‘test/unit’

class MyTest < Test::Unit::TestCase

end
```

```
require ‘test/unit’

class MyTest < Test::Unit::TestCase

  def test_method_is_called

  end

  def test_another_method_is_called

  end

end
```

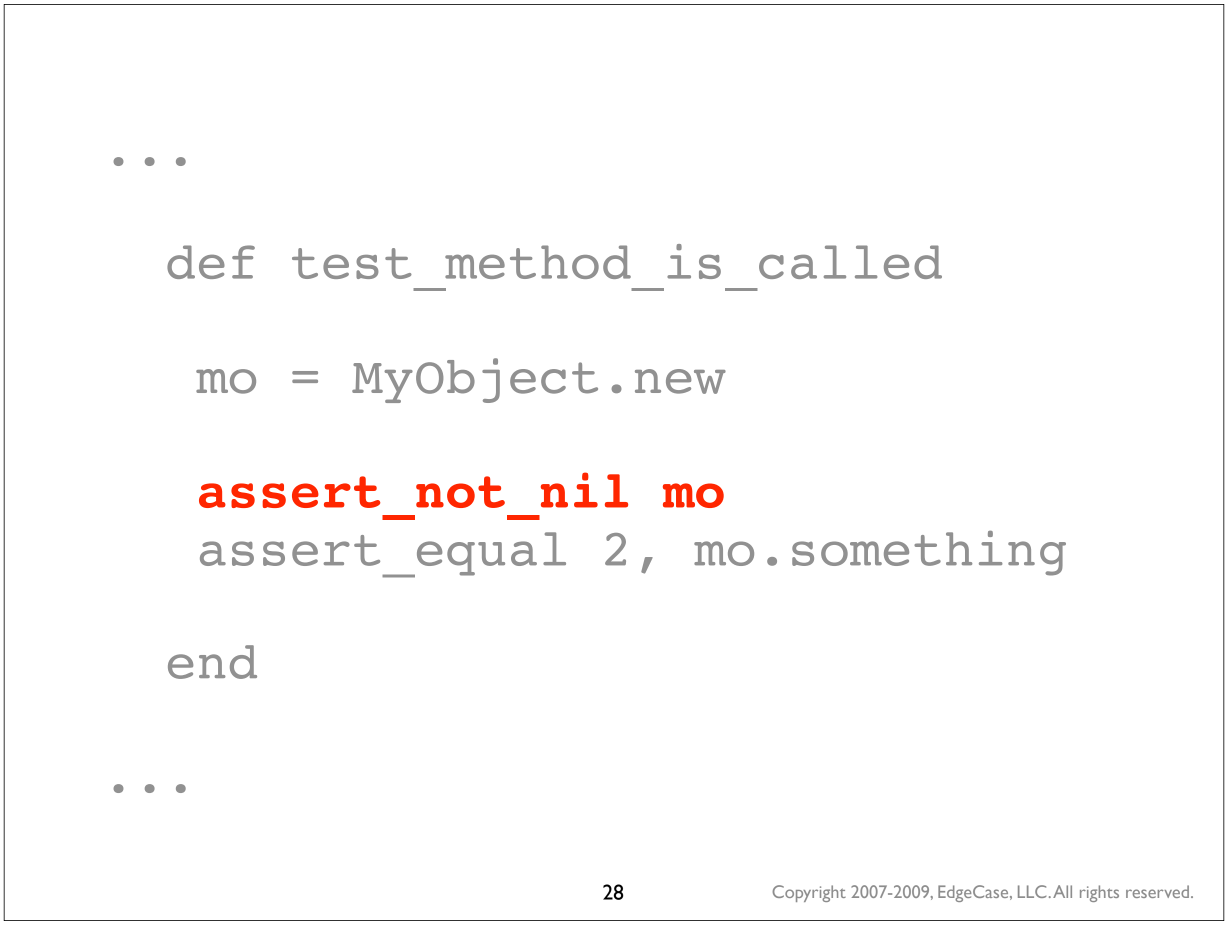
```
require ‘test/unit’

class MyTest < Test::Unit::TestCase

  def test_method_is_called

  end

  def test_another_method_is_called

  end

end
```

```
require ‘test/unit’

class MyTest < Test::Unit::TestCase

  def test_method_is_called

  end

  def test_another_method_is_called

  end

end
```

```
...

  def test_method_is_called

   mo = MyObject.new

   assert_not_nil mo
   assert_equal 2, mo.something

  end

...
```



```
...

  def test_method_is_called

   mo = MyObject.new

   assert_not_nil mo
   assert_equal 2, mo.something

  end

...
```



```
...

  def test_method_is_called

   mo = MyObject.new

   assert_not_nil mo
   assert_equal 2, mo.something

  end

...
```

```
...

  def test_method_is_called

   mo = MyObject.new

   assert_not_nil mo
   assert_equal 2, mo.something

  end

...
```

assertions

```
assert(truth, error_message=nil)
assert_block(failed_message=”assert block failed.”) {}
assert_equal(expected, actual_value, error_message=nil)
assert_not_equal(expected, actual_value, error_message=nil)
assert_in_delta(expected_float, actual_float, delta, message=’‘)
assert_instance_of(clazz, object, message=’‘)
assert_kind_of(clazz, object, message=”“)
assert_match(regex_pattern, string, message=’‘)
assert_no_match(regex_pattern, string, message=’‘)
assert_nil(nil_object, message=”“)
assert_not_nil(object, message=”“)
assert_same(object_one, object_two, message=”“)
assert_not_same(object_one, object_two, message=”“)
assert_operator(object_one, operator, object_two, message=”“)
assert_send(send_array, message=’‘)
assert_respond_to(object, message, failure_message=”“)
assert_raise(args) {...}
assert_nothing_raised(args) {}
assert_throws(expected_error_as_symbol, failure_message, &proc)
assert_nothing_thrown(error_message=’’, &proc)
flunk(message=”Flunked”)
```

```
assert(truth, error_message=nil)

assert_equal(expected, actual_value, error_message=nil)
assert_not_equal(expected, actual_value, error_message=nil)

assert_match(regex_pattern, string, message=’‘)

assert_nil(nil_object, message=”“)
assert_not_nil(object, message=”“)
```

assert

```
assert(truth,error_message=nil)
```

```
assert [].empty?
assert [2,4,6].include?(4)
```

error messages

```
assert(truth,error_message=nil)
```

```
assert [2,4,6].empty?
```

```
$ ruby assert_message_test.rb
Loaded suite assert_message_test
Started
F
Finished in 0.054849 seconds.

  1) Failure:
test_messages(AssertMessageTest)
[assert_message_test.rb:6]:
<false> is not true.

1 tests, 1 assertions, 1 failures, 0 errors
```

```
$ ruby assert_message_test.rb
Loaded suite assert_message_test
Started
F
Fini

  1) Failure
test_messages(AssertMessageTest)
[assert_message_test.rb:6]:
<false> is not true.

1 tests, 1 assertions, 1 failures, 0 errors
```

<false> is not true.

```
$ ruby assert_message_test.rb
Loaded suite assert_message_test
Started
F
Fini

  1) Failure
test_messages(AssertMessageTest)
[assert_message_test.rb:6]:
<false> is not true.

1 tests, 1 asser
```

<false> is not true.

not helpful!

error messages

`assert(truth,`**`error_message=nil`**`)`

```
assert [2,4,6].empty?,
   "Expected the array to be
empty but it is not"
```

error messages

```
assert(truth,error_message=nil)
```

```
assert [2,4,6].empty?,
   "Expected the array to be
empty but it is not"
```

```
$ ruby assert_message_test.rb
Loaded suite assert_message_test
test_messages(AssertMessageTest)
[assert_message_test.rb:6]:
Expected the array to be empty but it is not.
<false> is not true.

1 tests, 1 assertions, 1 failures, 0 errors
```

Expected the array to be empty but it is not.

```
$ ruby assert_message_test.rb
Loaded suite assert_message_test
test_messages(AssertMessageTest)
[assert_message_test.rb:6]:
Expected the array to be empty but it is not.
<false> is not true

1 tests, 1 asser
```

Expected the array to be empty but it is not.

much better!

more assertions

assert_not_nil

```
assert_not_nil(non_nil_object,
               error_message=nil)
```

```
assert_not_nil [1,5]
assert_not_nil Object.new
```

assert_equal

```
assert_equal(expected, actual,
        error_message=nil)
```

```
assert_equal "foo", :foo.to_s
assert_equal 2, [2,4,6].first
```

assert_not_equal

```
assert_not_equal(expected,
     actual, error_message=nil)
```

```
assert_not_equal "bar", :foo.to_s
assert_not_equal 2, [2,4,6].last
```

assert_match

```
assert_match(pattern, string,
     error_message=nil)
```

```
assert_match /\d/, "w00t"
assert_match /not nil/,
            e.error_message
```

assert_match

```
assert
      e
```

great for
substring matching

```
assert
assert_match /not nil/,
             e.error_message
```

```
assert_match(pattern, string, message='')

assert_match /\d/, "s0mething"
```

Test Driven Development with Rails

Red / Green / Refactor

Different Names

~~Test~~

~~First~~

~~Design~~

Different Names

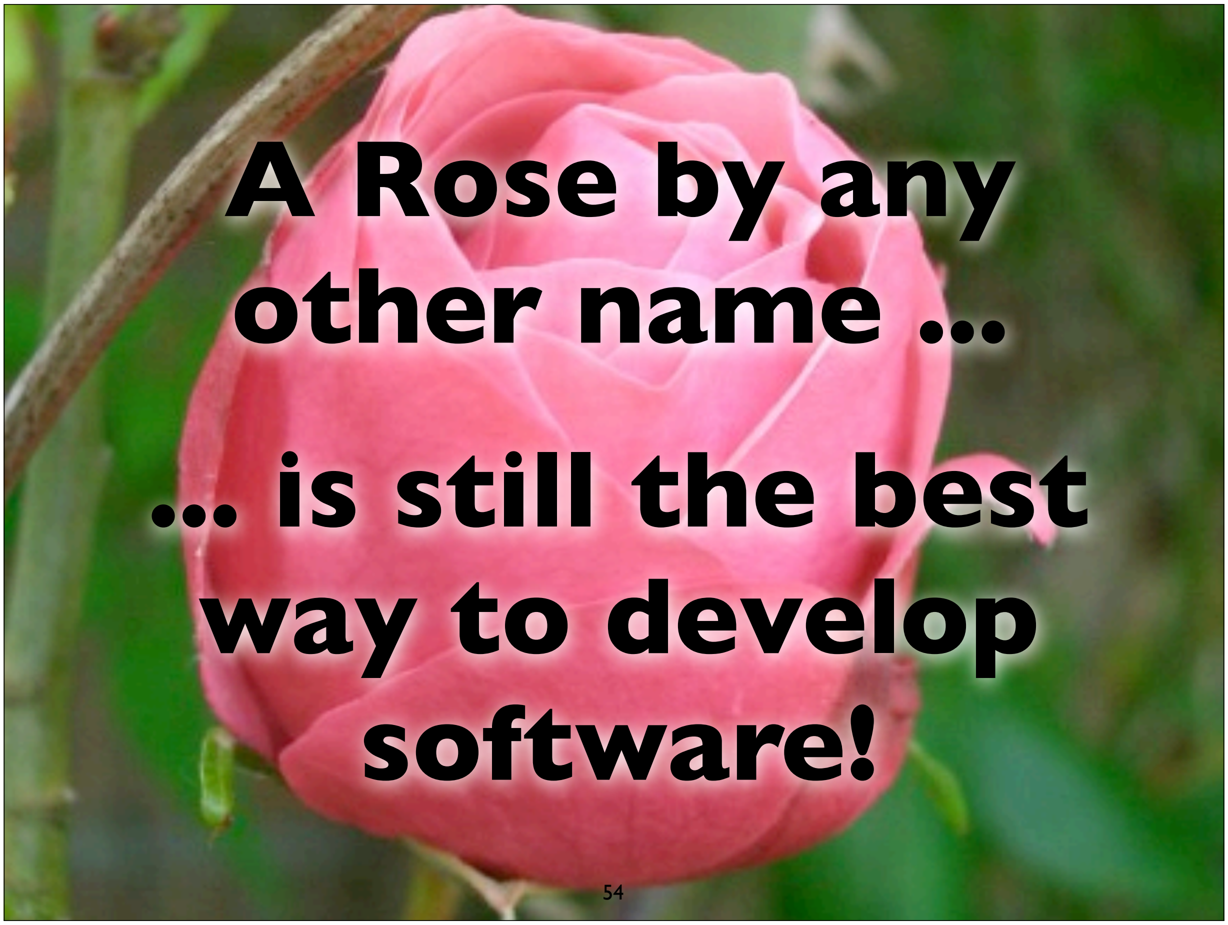
~~Test~~
~~First~~
~~Development~~

Different Names

~~Test~~
~~Driven~~
~~Development~~

Different Names

Behavior Driven Development

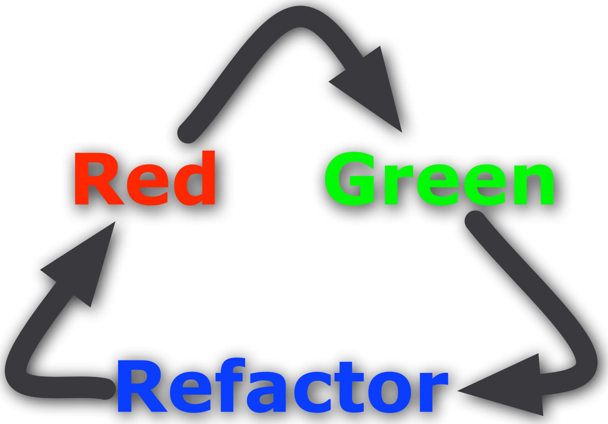


A Rose by any other name ...

... is still the best way to develop software!

Red
Green
Refactor

Write just enough code for the test to pass.



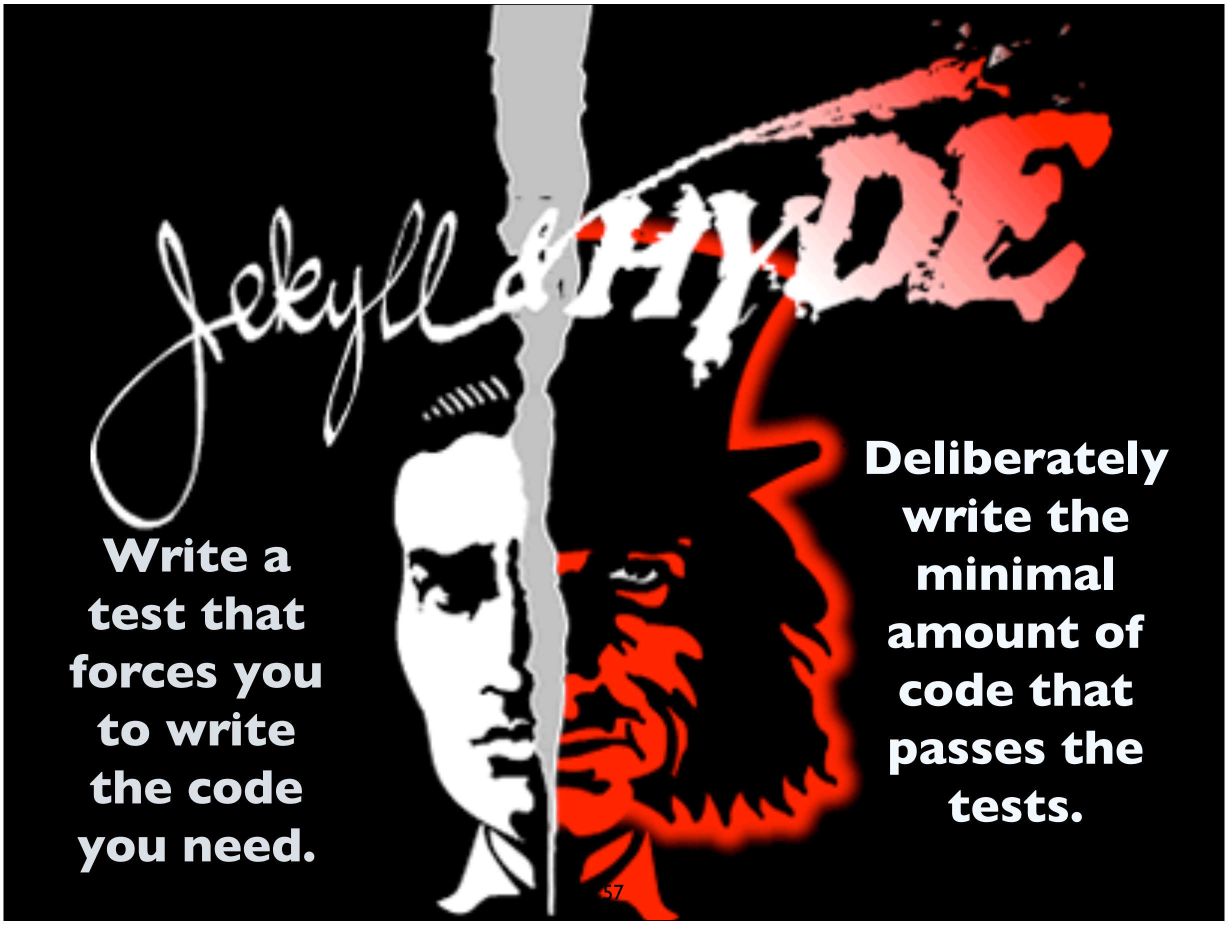
Red

Green

Refactor

Write a new test

Remove duplication, clean up, etc



Write a test that forces you to write the code you need.

Deliberately write the minimal amount of code that passes the tests.

DEMO

example in `01_ping_pong_demo`

General Principles

General Principles

Is it OK to have failing Unit Tests?

Only during the RED phase

Unit tests should always pass 100% on checked in code!

General Principles

Any Special Conventions for Testing?

```
def test_something
  # Given (setup)
  [... fixture data, mock expectations ...]

  # When
  [... run the code to be tested ...]

  # Then
  [... assert that the results are expected ...]
end
```

```
def test_popping_a_stack_returns_last_item_pushed
  # Given
  stack = Stack.new
  stack.push(:first)
  stack.push(:last)

  # When
  item = stack.pop

  # Then
  assert_equal :last, item
end
```

General Principles

What if my code uses other modules?

Use Mock Objects

- FlexMock
- Mocha
- Schmock

```
def test_acquire_token_with_bad_user
  # Given
  @login_service.
    should_receive(:authenticate).
    with(“user”, “password”).and_return(nil)

  # When
  token =
    token_service.acquire(“user”, “pw”, “app”)

  # Then
  assert_nil token
end
```

General Principles

Should we have one test per method?
And one TestCase per class?

NO!

Organize your tests around
behavior,
not structure.

```
def test_acquire_token_with_good_user
def test_acquire_token_with_bad_user
def test_release_an_acquired_token
def test_release_an_already_released_token
def test_release_a_nonexisting_token
def test_release_an_expired_token
def test_find_by_token_string
```

General Principles

Who Should Run the Tests?

No matter who runs them normally ...

All tests should be runnable by *development*

General Principles

How long should the tests take to run?

As Fast As Possible

Survey

- How long does it take to run your unit tests?
- How often do you run your unit tests?

Jeff Nielsen

The Psychology of Build Times

- Unit Tests: **<10 Seconds**
- Checkin Tests: **<10 Minutes**

Three Tiers of Tests
(1) Unit Tests

- Runs in seconds
- Developer run (often!)
- Developed via Red/Green/Refactor

Three Tiers of Tests
(2) Check-in Tests

- Runs in minutes
- Run at Check-in time
- Integration Scope

Three Tiers of Tests
(3) Nightly Tests

- (possibly) Long Running
- Automatic and Off-line
- Functional Scope (requirements & load)

General Principles

So, what counts as a unit test?

Not a Unit Test if ...

- Talks to the database
- Communicates across the network
- Touches the file system
- Can’t run concurrently with other unit tests
- Requires some manual setup

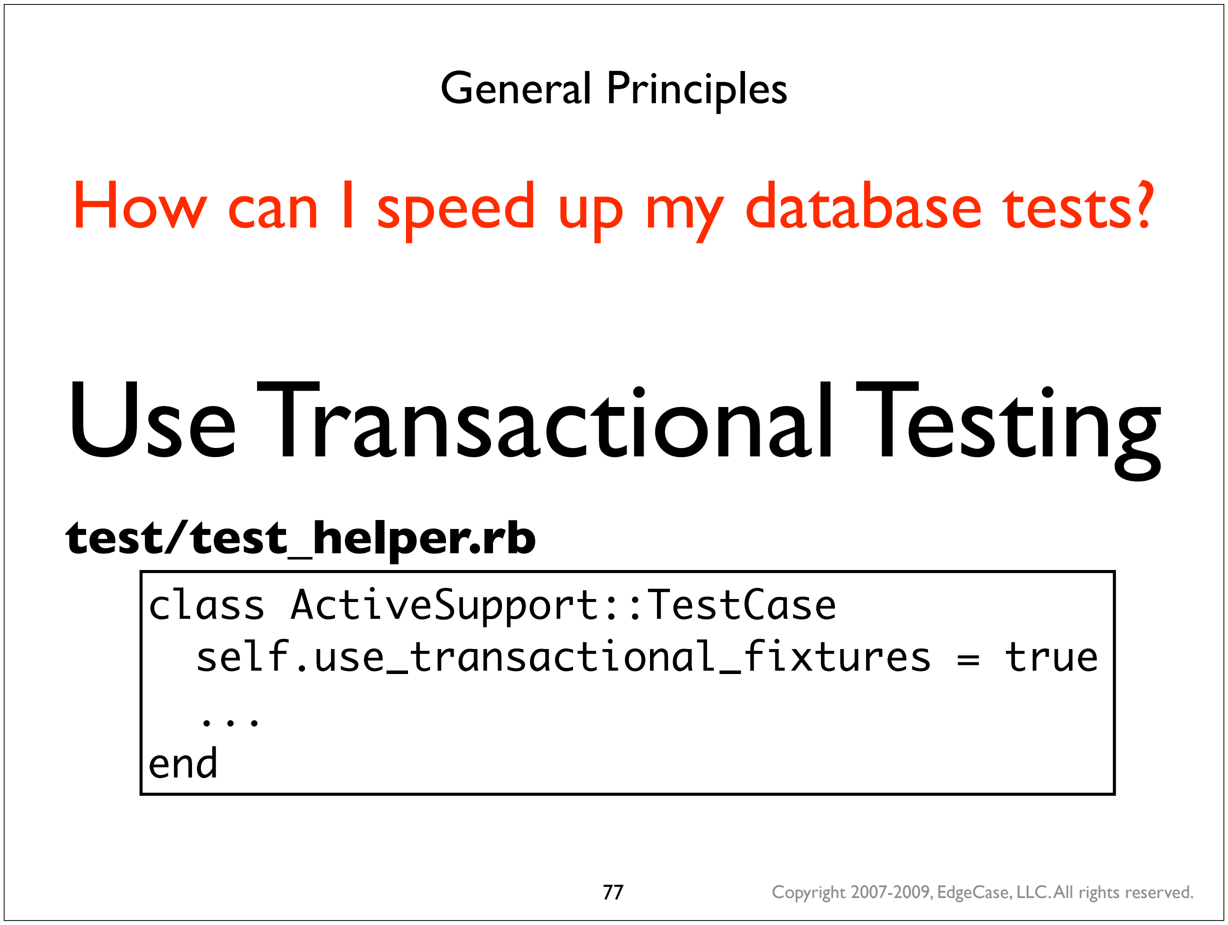
-- Michael Feathers

General Principles

What if bugs are found in the software?

Add a test to check for the bug.

- Prevents bug regression
- Tests should be entered as high up the testing chain as is feasible.



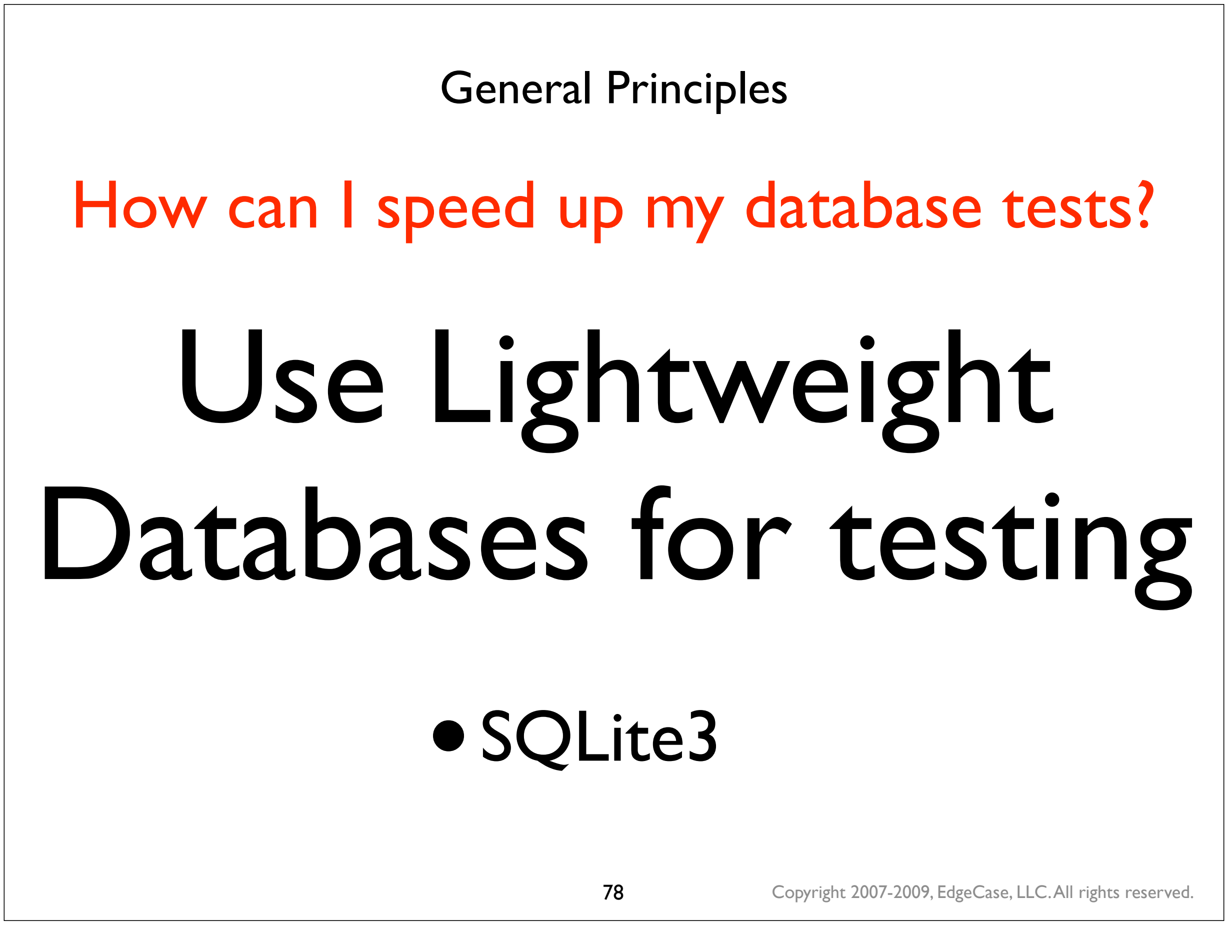
General Principles

How can I speed up my database tests?

Use Transactional Testing

test/test_helper.rb

```
class ActiveSupport::TestCase
  self.use_transactional_fixtures = true
  ...
end
```



General Principles

How can I speed up my database tests?

Use Lightweight Databases for testing

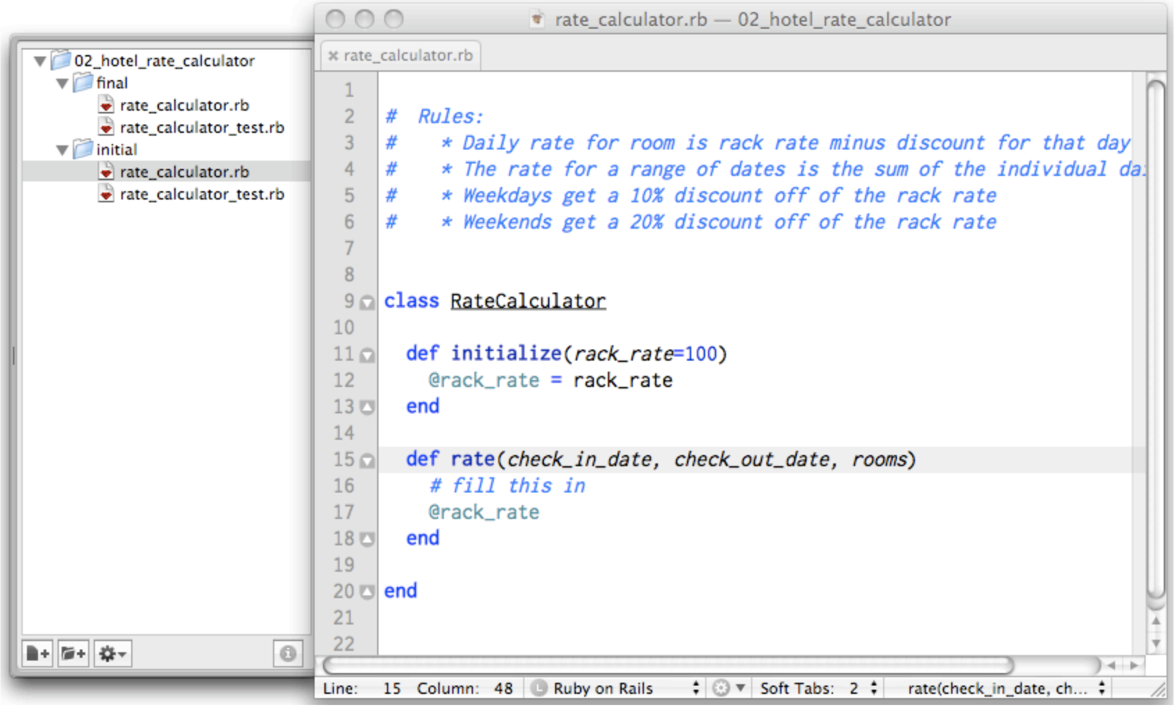
- SQLite3

Questions?

Summary

- Keep Unit Tests Fast
- Run Tests Often
- Use Red/Green/Refactor

Lab 2
Hotel Rate Calculator



```ruby

#  Rules:
#    * Daily rate for room is rack rate minus discount for that day
#    * The rate for a range of dates is the sum of the individual da
#    * Weekdays get a 10% discount off of the rack rate
#    * Weekends get a 20% discount off of the rack rate


class RateCalculator

  def initialize(rack_rate=100)
    @rack_rate = rack_rate
  end

  def rate(check_in_date, check_out_date, rooms)
    # fill this in
    @rack_rate
  end

end


```

Line: 15 Column: 48 Ruby on Rails Soft Tabs: 2 rate(check_in_date, ch...

Lab 2: Hotel Rate Calculator

- The daily rate for a room is its rack_rate minus the daily discount.
- The rate for a range of dates is the sum of the individual daily rates.
- Dates during the week get a 10% discount off of the rack rate.
- Dates on weekends get a 20% discount off of the rack rate.

Lab 2: Hotel Rate Calculator

Extra Credit Rules:

- 10-49 rooms get an extra 20% off the rack_rate.
- 50+ rooms get an extra 30% off the rack_rate.
- At no time should the combined discount exceed 40%.

Test Driven Development with Rails

RSpec

a new way of doing things right

BDD

If you are testing, you are not wrong.

TDD

common problems

What is a unit?

Where do I start?

What is the intent?

How much is enough?

design by accident

What do we want?

avoid testing state

specify not verify

make it easier to test more effectively

reveal our intentions

focus on the design

focus on behavior

RSpec

rspec.info

```
gem install rspec
```

forget units,
describe the contexts

```
describe Ring, “when empty” do ...

describe Ring, “with one item” do ...

describe Ring, “when full” do ...
```

one block per context

```
describe Ring, 'when being created' do


end
```

what should happen?

```
describe Ring, 'when being created' do

  it 'should give size’
  it 'should indicate it is empty'
  it 'should return length of zero'

end
```

```
$ spec ring_spec.rb
PPP

Finished in 0.006374 seconds

3 examples, 0 failures, 3 pending

Pending:
Ring when being created should give size (Not Yet
Implemented)
Ring when being created should indicate it is empty
(Not Yet Implemented)
Ring when being created should return length of zero
(Not Yet Implemented)
```

```
$ spec ring_spec.rb
PPP

Finished in 0.006374 seconds

3 examples, 0 failures, 3 pending

Pending:
Ring when being created should give size (Not Yet
Implemented)
Ring when being created should indicate it is empty
(Not Yet Implemented)
Ring when being created should return length of zero
(Not Yet Implemented)
```

```
...

it 'should give size’ do
  number = 3
  ring = Ring.new(number)

  ring.size.should == number
end

...
```

```
$ spec ring_spec.rb
.PP

Finished in 0.006735 seconds

3 examples, 0 failures, 2 pending

Pending:
Ring when being created should indicate it is ...
Ring when being created should return length ...
```

```
...

it 'should give size’ do
  number = 3
  ring = Ring.new(number)

  ring.size.should == number
end

...
```

specifications

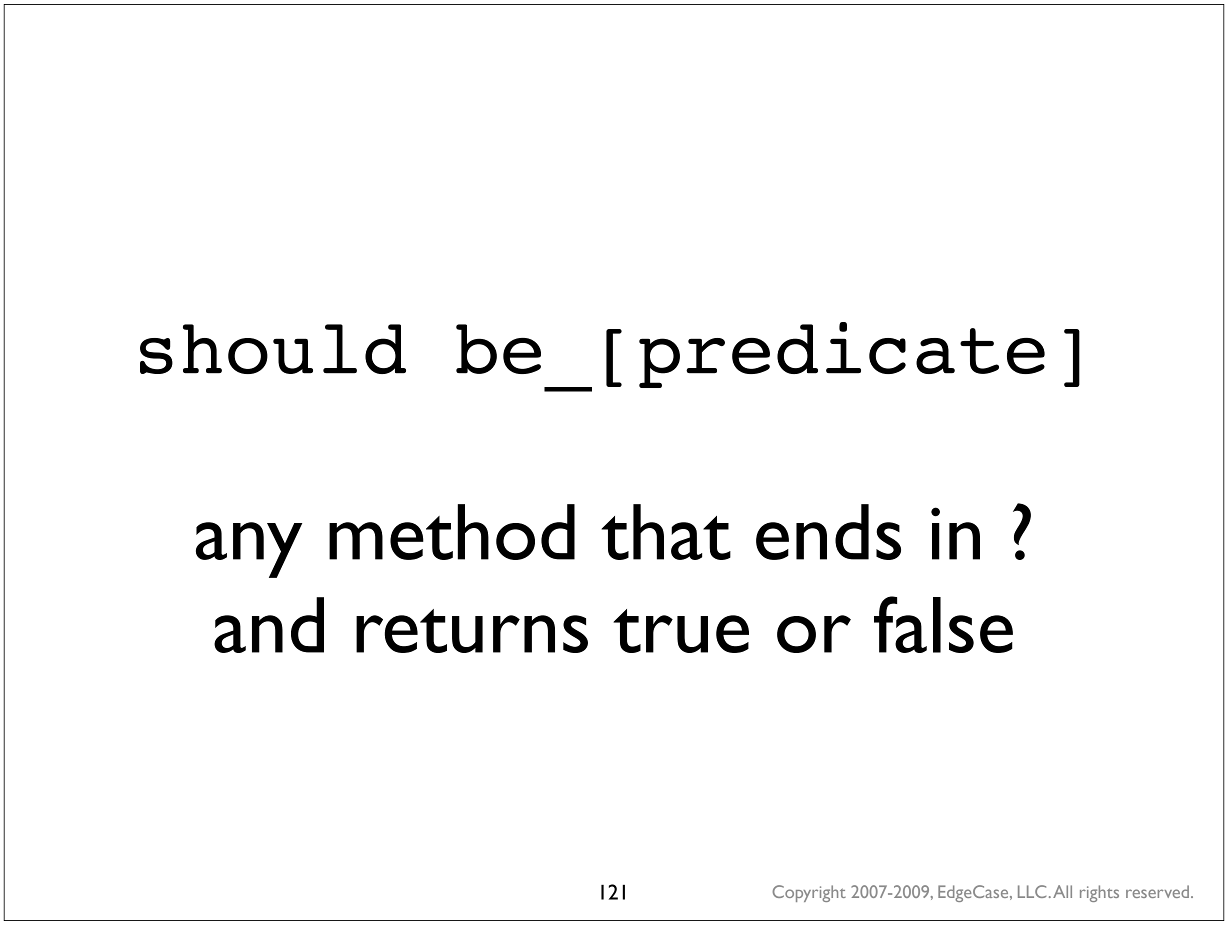
```
should(matcher=nil)

should_not(matcher=nil)
```

matchers

two types

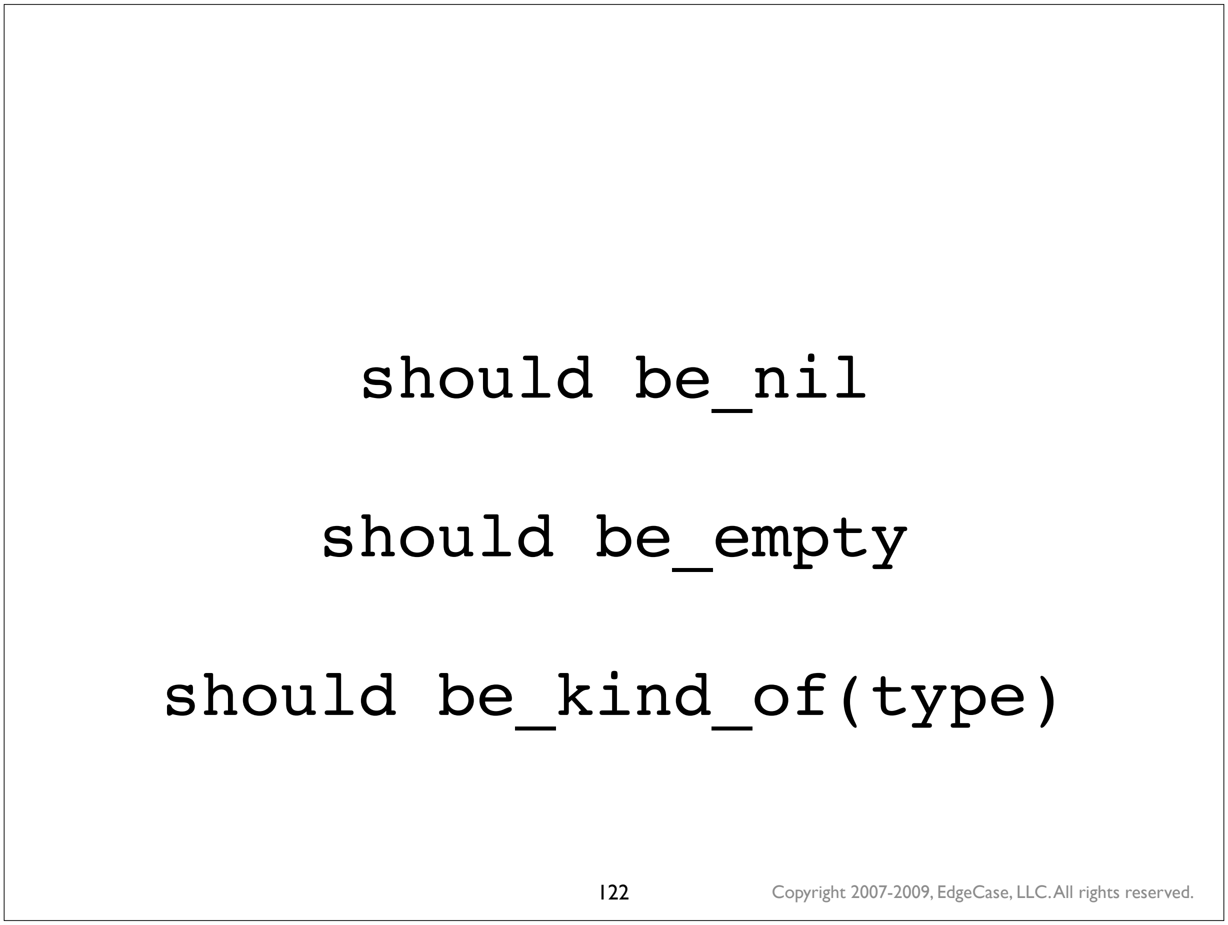
- predicates
- custom matchers

predicates



`should be_[predicate]`

any method that ends in ?
and returns true or false



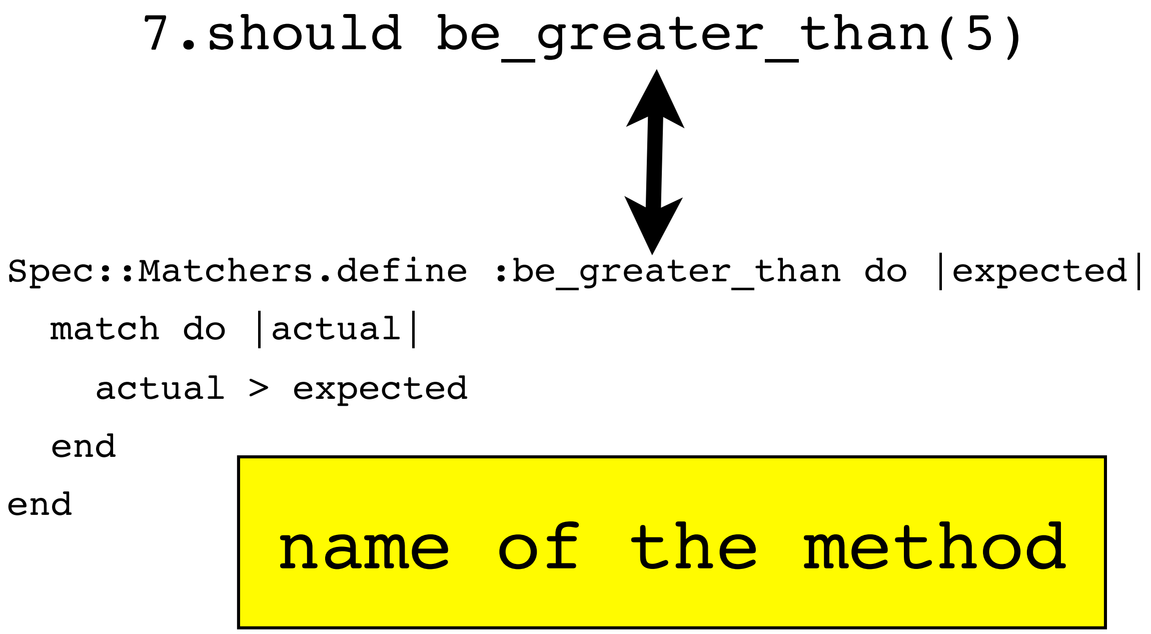
```
should be_nil

should be_empty

should be_kind_of(type)
```

custom matchers

```
describe 7 do
  it "should be greater than 5" do
    7.should > 5
  end
end
```

```
describe 7 do
  it "should be greater than 5" do
    7.should be_greater_than(5)
  end
end
```

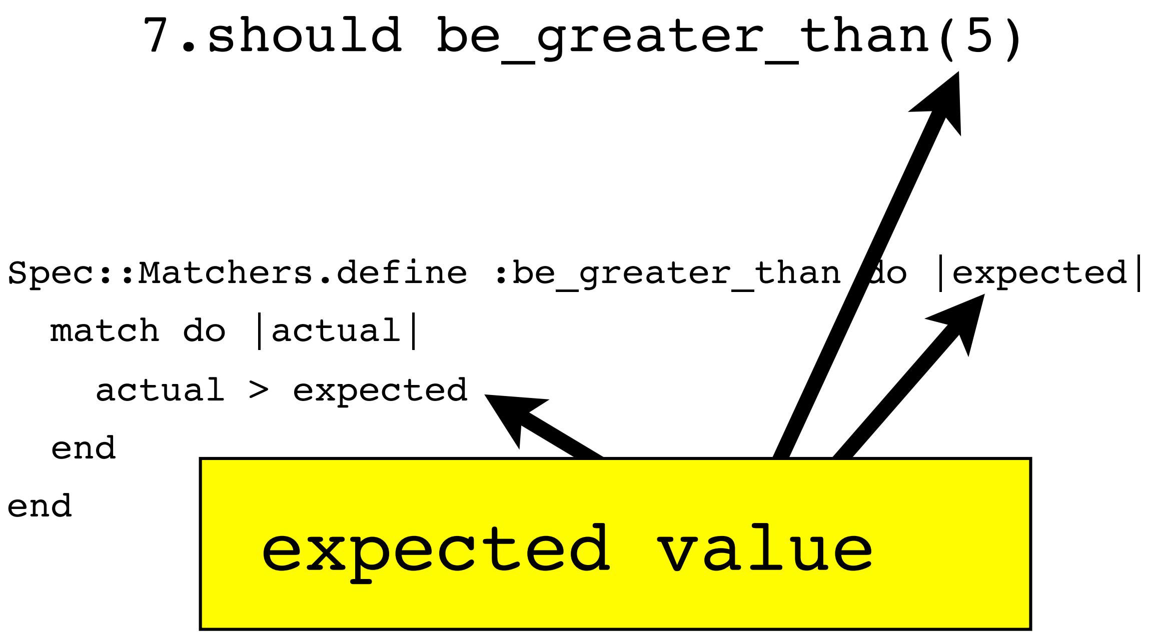
```
Spec::Matchers.define :be_greater_than do |expected|
  match do |actual|
    actual > expected
  end
end
```



```
7.should be_greater_than(5)
```

```
Spec::Matchers.define :be_greater_than do |expected|
  match do |actual|
    actual > expected
  end
end
```

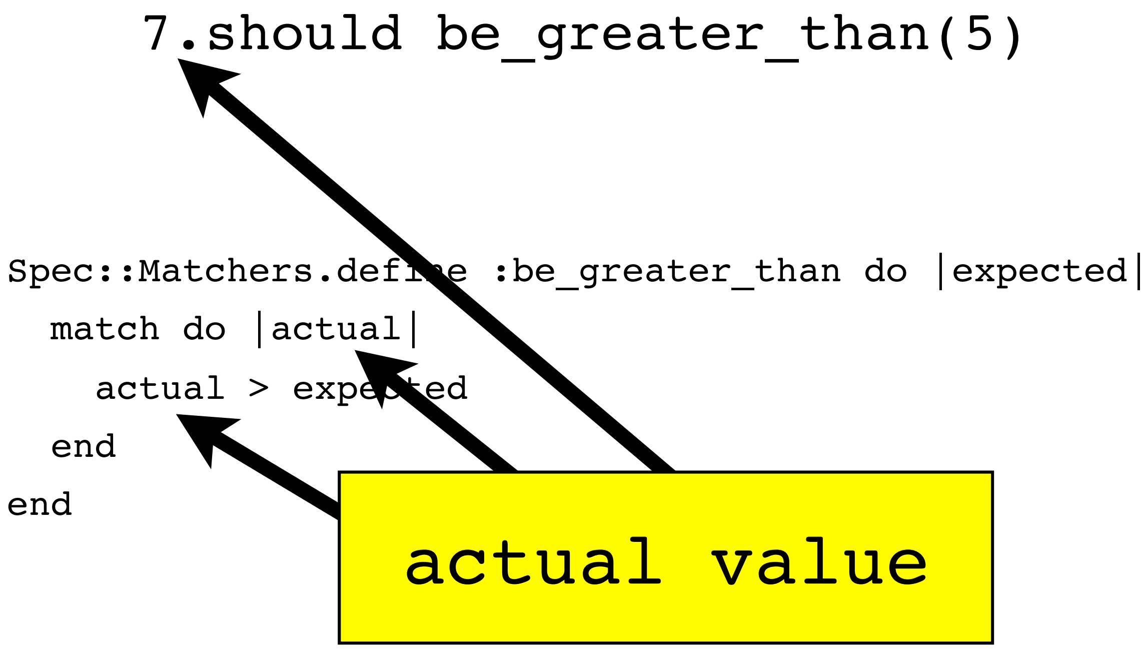
name of the method



```
7.should be_greater_than(5)
```

```
Spec::Matchers.define :be_greater_than do |expected|
  match do |actual|
    actual > expected
  end
end
```

expected value



```
7.should be_greater_than(5)
```

```
Spec::Matchers.define :be_greater_than do |expected|
  match do |actual|
    actual > expected
  end
end
```

```
describe 7 do
  it "should be greater than 5" do
    7.should be_greater_than(5)
  end
end

# Auto generate the description
describe 7 do
  it { should be_greater_than(5)}
end
```

rspec-on-rails-matchers

```
post.should have_many(:comments)

user.should validate_presence_of(:name)

object.should observe(:model)

response.should have_submit_button
```

others

Wiki on rspec github page:

http://github.com/joshknowles/rspec-on-rails-matchers/
http://github.com/pd/rspec_hpricot_matchers/
http://github.com/thoughtbot/shoulda/
http://github.com/carlosbrando/remarkable/
http://github.com/technoweenie/rspec_on_rails_on_crack/

time to play!

Test Driven Development with Rails

Lab 3: Implement a Ring

Lab 3: Implement a Ring

Implement a ring using the outline in:

```
ring_spec.rb
```

- ✓ Drive the implementation using RSpec
- ✓ See if you can get to 100% coverage

Test Driven Development with Rails

Shoulda

Giving Tests Context

BDD vs TDD Controversy

BDD => RSpec

Cool Features in RSpec

- Nested Contexts for testing
- Specialized Matcher Syntax (obj.should)

Downsides to RSpec

- Major departure from Test::Unit
- Lots of "magic" involved in running specs
    - magic exposed during debugging/extending
- Slow
- Invades the global namespace
- API changes over time
    - (context/should VS describe/it)

Test::Unit is pretty good

(can’t we just find a way of using it?)

Shoulda

Shoulda Features

- Runs on top of Test::Unit
- Uses Test::Unit asserts
- Nested contexts (with context/should)
- Very lightweight

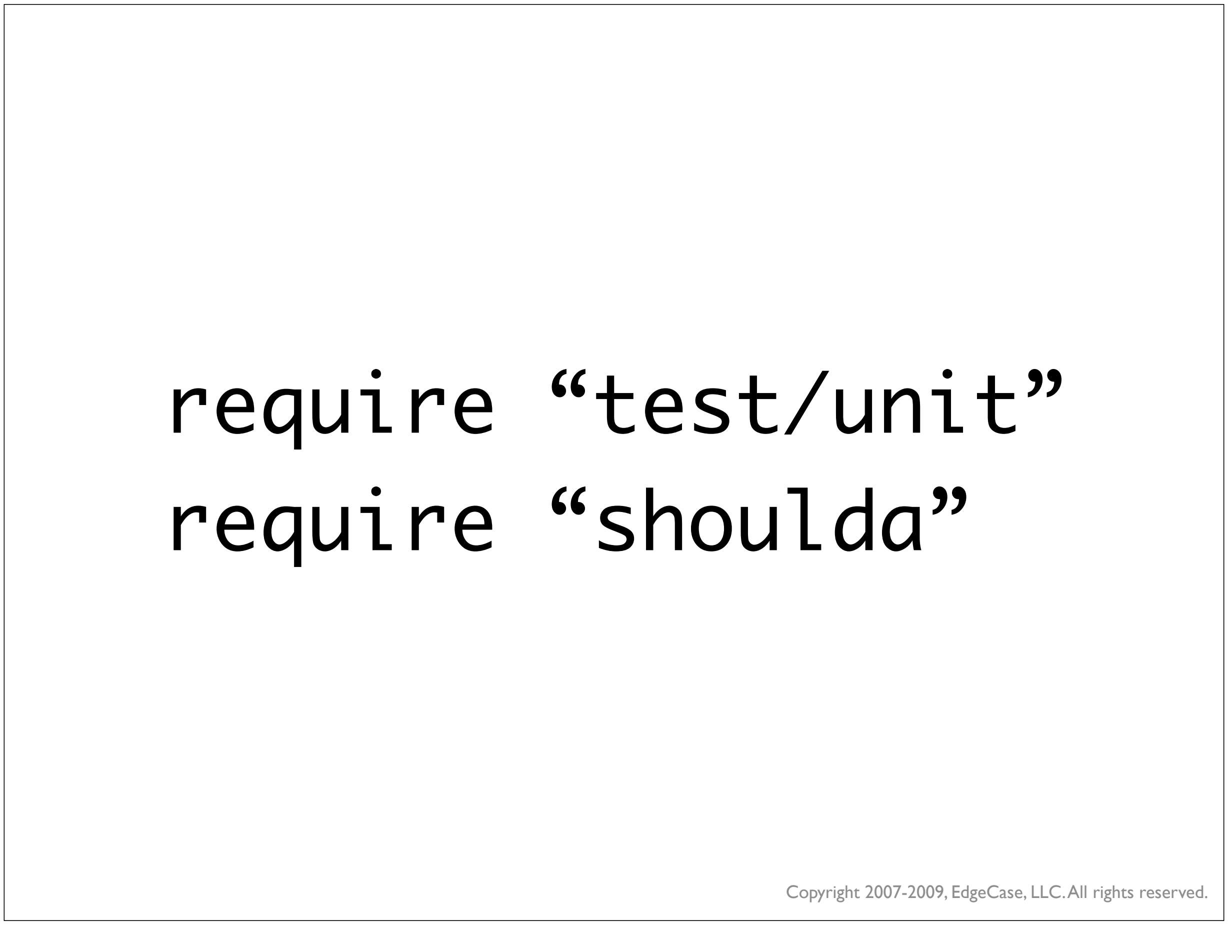
Shoulda Features

- No ...
    - Global namespace invasion
    - No fancy matchers

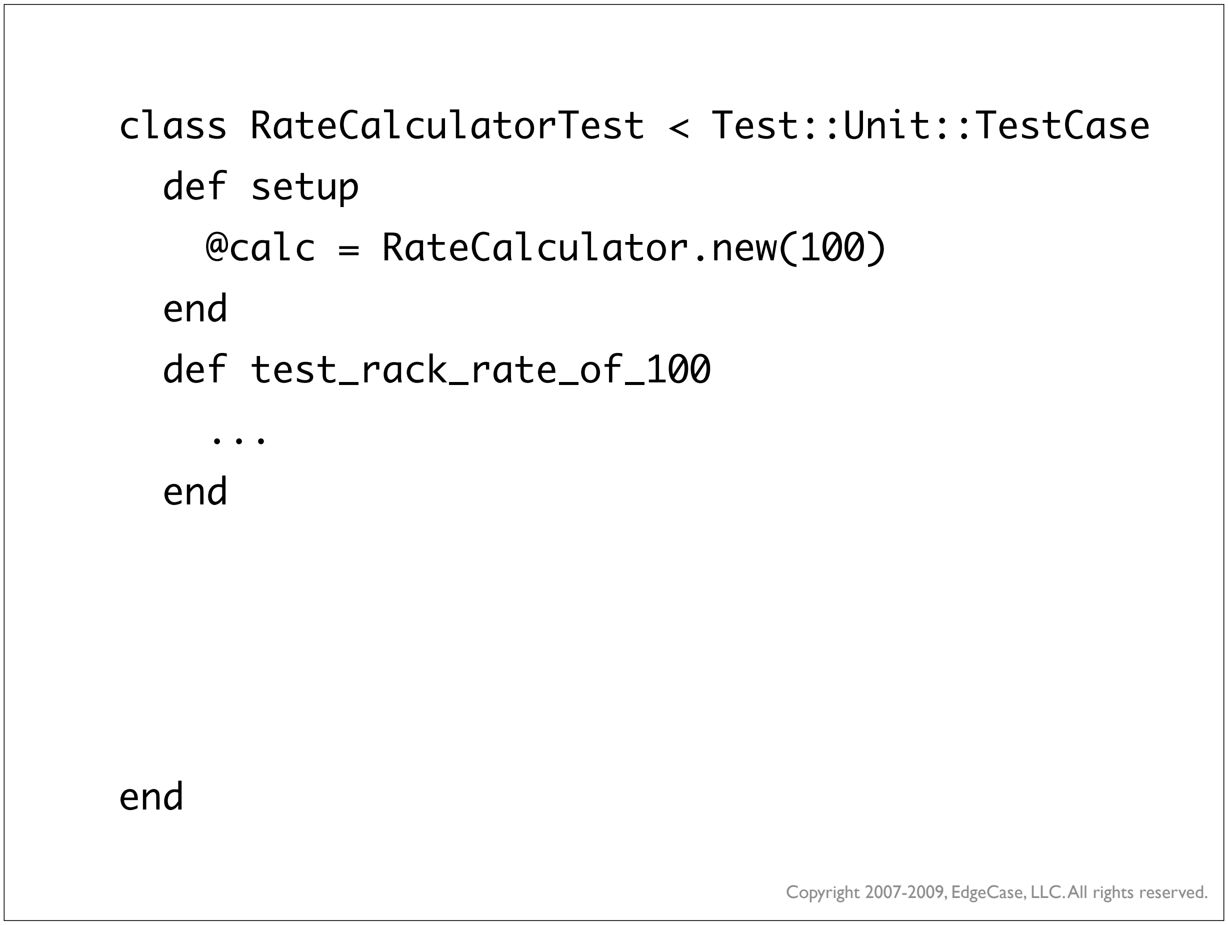
Shoulda/Rails

- When testing Rails, Shoulda provides a number of rails specific “macros” to test common situations.
- More on this when we get to Rails testing.

Shoulda in Action



```
require “test/unit”
require “shoulda”
```



```
class RateCalculatorTest < Test::Unit::TestCase
  def setup
    @calc = RateCalculator.new(100)
  end
  def test_rack_rate_of_100
    ...
  end




end
```

```
class RateCalculatorTest < Test::Unit::TestCase
  def setup
    @calc = RateCalculator.new(100)
  end
  def test_rack_rate_of_100
    ...
  end
  def test_rack_rate_of_50
    @calc = RateCalculator.new(50)
    ...
  end
end
```

```
class RateCalculatorTest < Test::Unit::TestCase
  def setup
    @calc = RateCalculator.new(100)
  end
  def test_rack_rate_of_100
    ...
  end
  def test_rack_rate_of_50
    @calc = RateCalculator.new(50)
    ...
  end
end
```

Using Shoulda

```
class RateCalculatorTest < Test::Unit::TestCase
  context “A Rate Calculator” do
    context “with a rack rate of 100” do
      setup { @calc = RateCalculator.new(100) }
      should “get a 20% discount” do
        ...(test code)...
      end
    end
    context “with a rack rate of 50” do
      setup { @calc = RateCalculator.new(50) }
      should “get a 20% discount” do
        ...(test code)...
      end
    end
  end
```

```
class RateCalculatorTest < Test::Unit::TestCase
  context “A Rate Calculator” do
    context “with a rack rate of 100” do
      setup { @calc = RateCalculator.new(100) }
      should “get a 20% discount” do
        ...(test code)...
      end
    end
    context “with a rack rate of 50” do
      setup { @calc = RateCalculator.new(50) }
      should “get a 20% discount” do
        ...(test code)...
      end
    end
  end
```

```
class RateCalculatorTest < Test::Unit::TestCase
  context “A Rate Calculator” do
    context “with a rack rate of 100” do
      setup { @calc = RateCalculator.new(100) }
      should “get a 20% discount” do
        ...(test code)...
      end
    end
    context “with a rack rate of 50” do
      setup { @calc = RateCalculator.new(50) }
      should “get a 20% discount” do
        ...(test code)...
      end
    end
  end
```

```
class RateCalculatorTest < Test::Unit::TestCase
  context “A Rate Calculator” do
    context “with a rack rate of 100” do
      setup { @calc = RateCalculator.new(100) }
      should “get a 20% discount” do
        ...(test code)...
      end
    end
    context “with a rack rate of 50” do
      setup { @calc = RateCalculator.new(50) }
      should “get a 20% discount” do
        ...(test code)...
      end
    end
  end
```

```
class RateCalculatorTest < Test::Unit::TestCase
  context “A Rate Calculator” do
    context “with a rack rate of 100” do
      setup { @calc = RateCalculator.new(100) }
      should “get a 20% discount” do
        ...(test code)...
      end
    end
    context “with a rack rate of 50” do
      setup { @calc = RateCalculator.new(50) }
      should “get a 20% discount” do
        ...(test code)...
      end
    end
  end
```

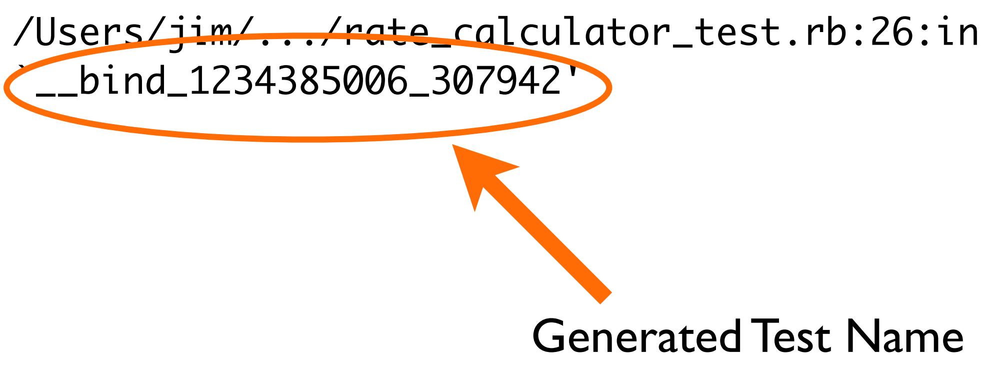
```
context 'A rate calculator' do
  context 'with a rack rate of 50' do
    context 'on a weekday' do
      should 'be discounted by 10 percent' do
```

```
  1) Failure: test: A rate calculator with a rack
rate of 50 on a weekday should be discounted by 10
percent. (RateCalculatorTest)
    [stack trace elided]:
Total should be 46.0 but was 45.0.
<46.0> and
<45.0> expected to be within
<0.01> of each other.
```

The Stack Trace

```
/Users/jim/.../rate_calculator_test.rb:26:in
`__bind_1234385006_307942'
```

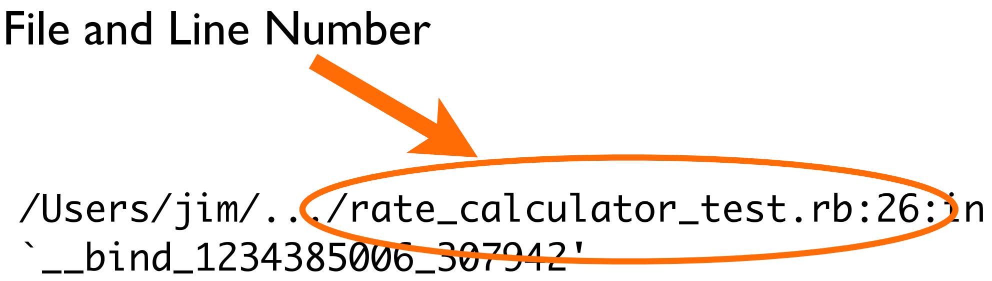
The Stack Trace



```
/Users/jim/.../rate_calculator_test.rb:26:in
`__bind_1234385006_307942'
```

Generated Test Name

The Stack Trace



File and Line Number

```
/Users/jim/.../rate_calculator_test.rb:26:in
`__bind_1234385006_307942'
```

Tips

Top Level Contexts

- Mention a noun
- Start with a capital letter

Nested Contexts

- Start with a connecting word
  - (e.g. on, with, when)
- Start with a lower case letter

The Should Block

- Should read with the “should” implied.

Read like a sentence

```
context 'A rate calculator' do
  context 'with a rack rate of 50' do
    context 'on a weekday' do
      should 'be discounted by 10 percent' do
```

A rate calculator with a rack rate of 50 on a weekday should be discounted by 10 percent.

Greed

Greed

- Greed is a dice game for two or more players using 5 six-sided dice
- Each player takes a turn consisting of several rolls
- The player decides whether to roll again, or not
- Each turn is scored and the points are added to the player's total score.

Scoring Rolls

- A triplet of any number is worth 100 * that number
- Except a triplet of ones, which are worth 1000 points
- Single "ones" are worth 100 points
- Single "fives" are worth 50 points
- All other values do not contribute points to the roll.

Rolling Again

- If a roll scored points, those points are added to the turns points, and the player may roll again with any non-scoring dice.
- If all dice have scored, then the player may roll again with all five of the dice.

Rolling Again

- If a roll does not score any points, then the player loses his turn and loses any points from that turn. This is called going “bust”.
- If a player elects to stop rolling before going bust, the points accumulated in this turn are added to his total score.

Get in the Game

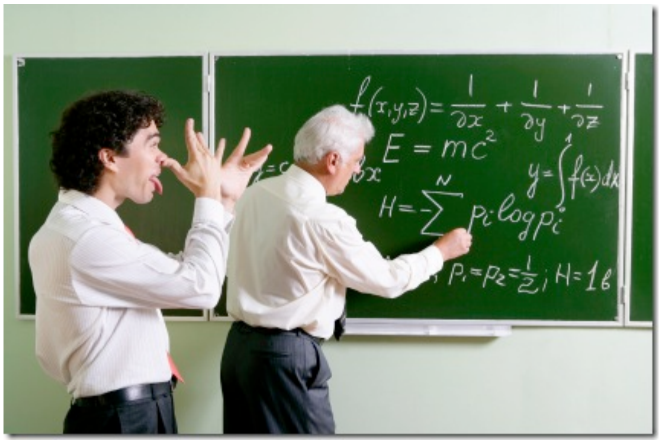
- A player must roll at least 300 in a single turn before he is allowed to start accumulating points.

Lab 4
Greed Scoring

LAB 4: Score Method

- Write a score method that calculates the number of points for a roll of dice.
  - score([1])   # => 100
  - score([5])   # => 50
  - score([3, 4, 2, 4, 4]) => 400
  - score([2, 1, 1, 3, 1]) => 1000

Test Driven Development with Rails

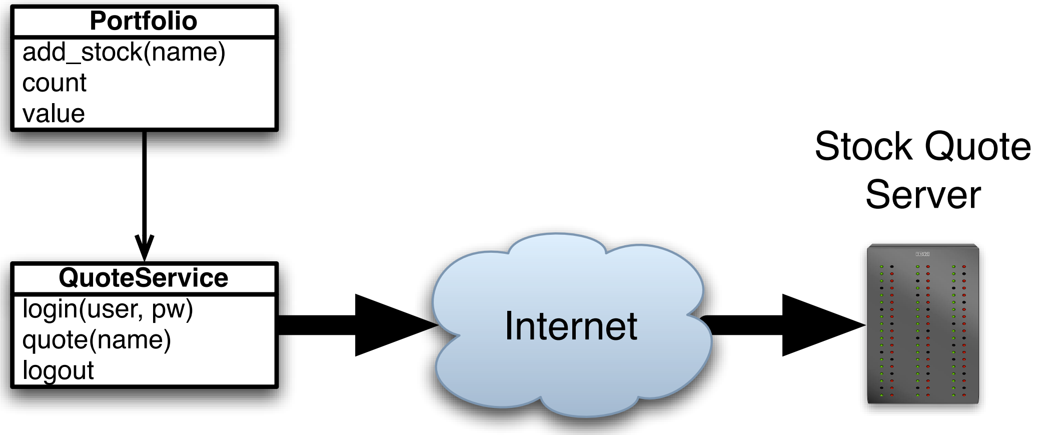

$f(x,y,z) = \frac{1}{\partial x} + \frac{1}{\partial y} + \frac{1}{\partial z}$
$E = mc^2$
$y = \int f(x)dx$
$H = -\sum^{N} p_i \log p_i$
$p_1 = p_2 = \frac{1}{2}; H = 1b$

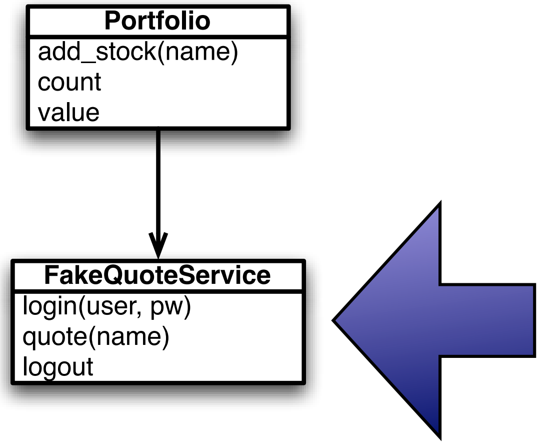
Mocking

Portfolio

- Tracks a number of stocks (by the company trading symbol, e.g. "APPL")
- Will accept new stock names to track
- Returns sum of all tracked stocks

- Quote Service





Portfolio
add_stock(name)
count
value

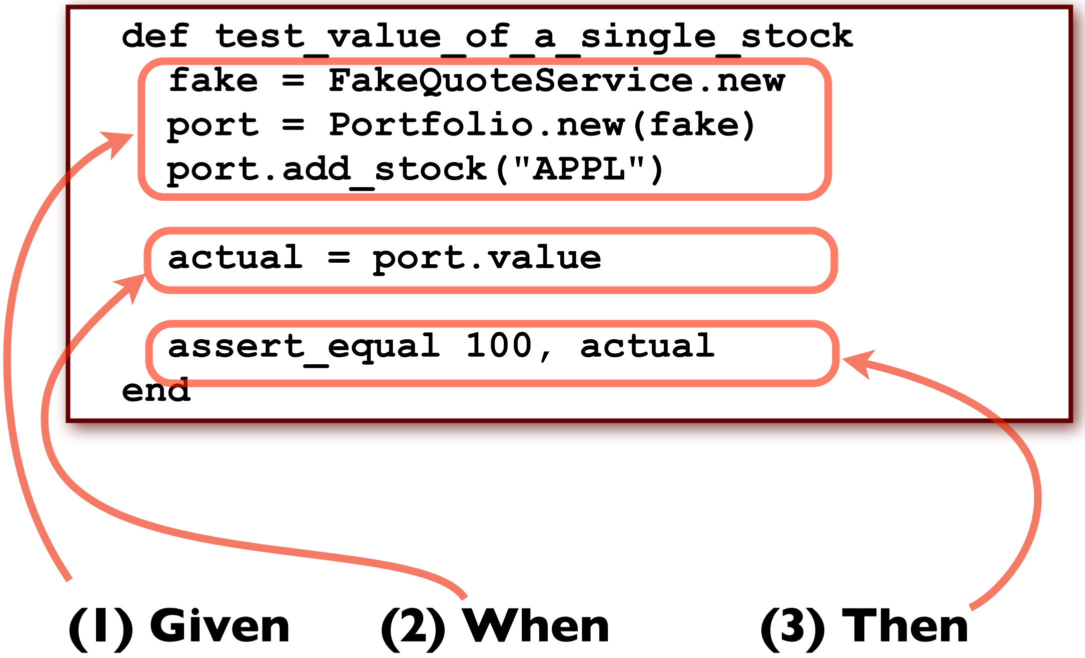
FakeQuoteService
login(user, pw)
quote(name)
logout

Terminology:

- Fake
- Test Double
- Mock
- Stub

Test Doubles

```
class FakeQuoteService
  def login
  end

  def quote(name)
    100
  end

  def logout
  end
end
```

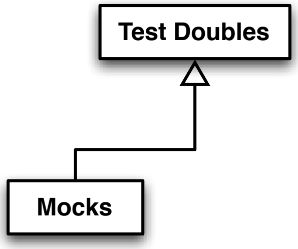


```
def test_value_of_a_single_stock
  fake = FakeQuoteService.new
  port = Portfolio.new(fake)
  port.add_stock("APPL")

  actual = port.value

  assert_equal 100, actual
end
```

(1) Given (2) When (3) Then

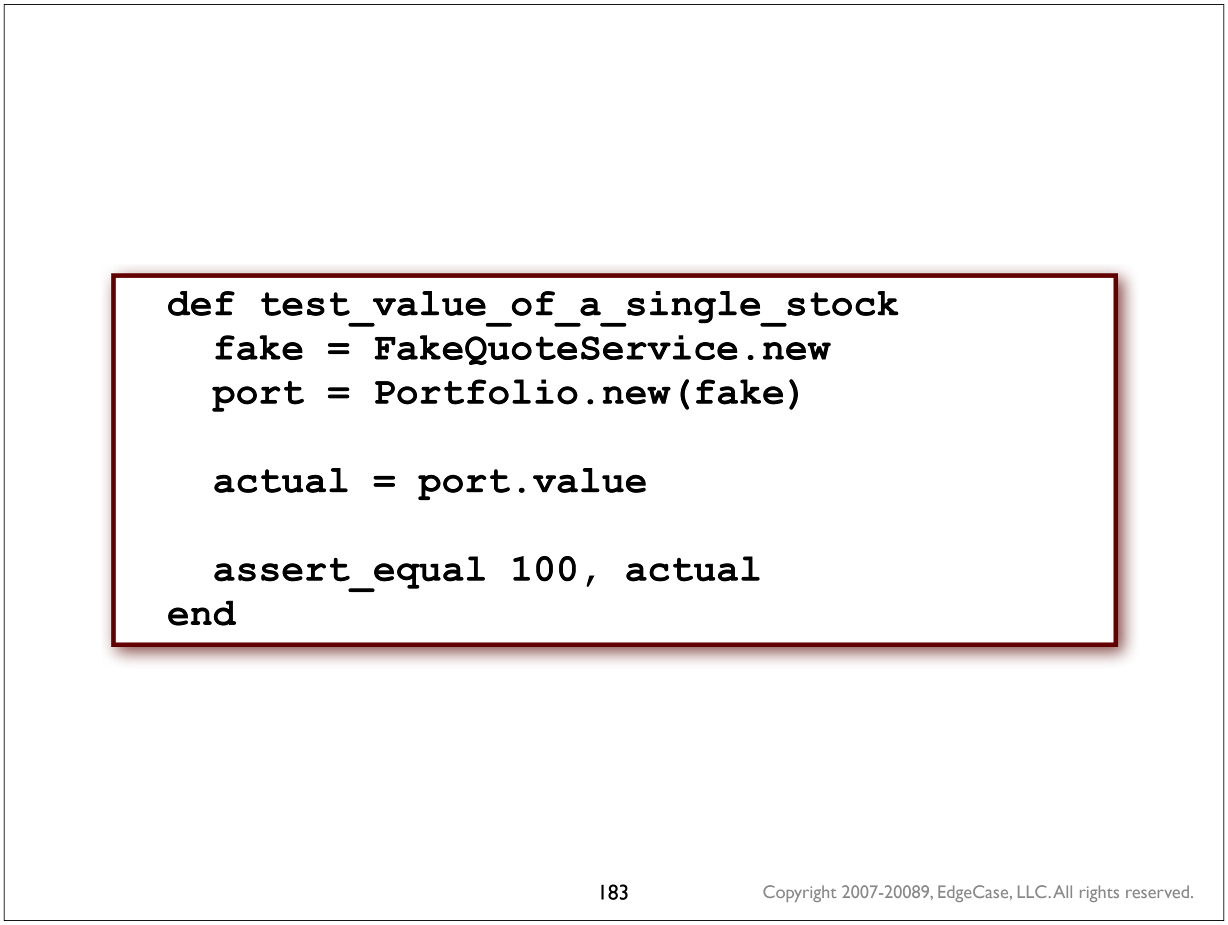
How Do You Handle

- More Stocks / Different Values?
- Validate the stock name is passed
- Log In/Out Requirements
  - Only call quote while logged in
  - Must logout when done

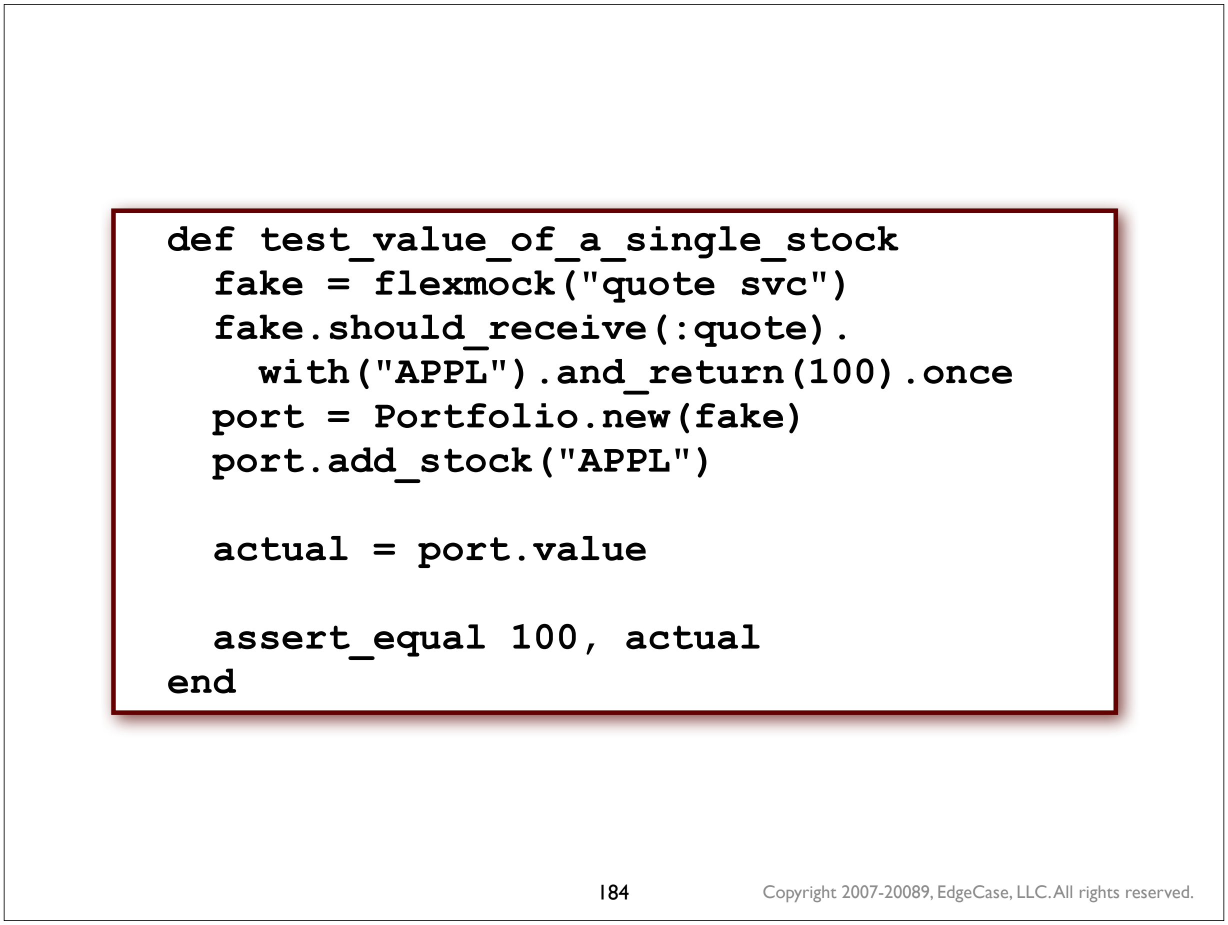


Test Doubles

Mocks



```
def test_value_of_a_single_stock
  fake = FakeQuoteService.new
  port = Portfolio.new(fake)

  actual = port.value

  assert_equal 100, actual
end
```



```
def test_value_of_a_single_stock
  fake = flexmock("quote svc")
  fake.should_receive(:quote).
    with("APPL").and_return(100).once
  port = Portfolio.new(fake)
  port.add_stock("APPL")

  actual = port.value

  assert_equal 100, actual
end
```

Create a Mock

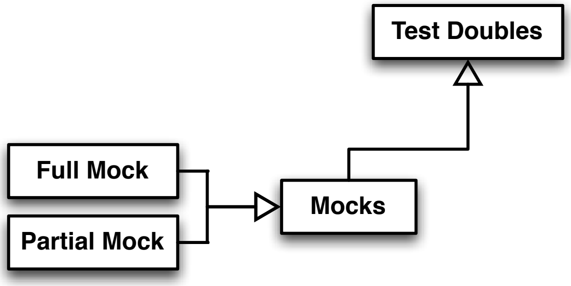
```
def test_value_of_a_single_stock
  fake = flexmock("quote svc")
  fake.should_receive(:quote).
    with("APPL").and_return(100).once
  port = Portfolio.new(fake)
  port.add_stock("APPL")

  actual = port.value

  assert_equal 100, actual
end
```

Configure a Mock

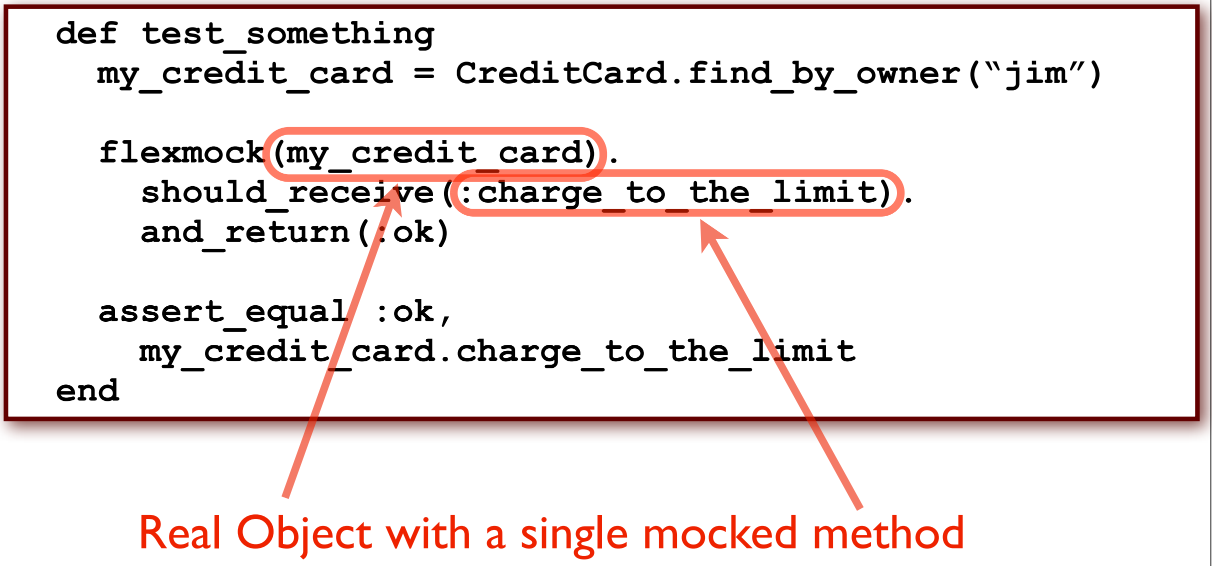
```
def test_value_of_a_single_stock
  fake = flexmock("quote svc")
  fake.should_receive(:quote).
    with("APPL").and_return(100).once
  port = Portfolio.new(fake)
  port.add_stock("APPL")

  actual = port.value

  assert_equal 100, actual
end
```

Use a Mock

```
def test_value_of_a_single_stock
  fake = flexmock("quote svc")
  fake.should_receive(:quote).
    with("APPL").and_return(100).once
  port = Portfolio.new(fake)
  port.add_stock("APPL")

  actual = port.value

  assert_equal 100, actual
end
```



Partial Mock



```
def test_something
  my_credit_card = CreditCard.find_by_owner(“jim”)

  flexmock(my_credit_card).
    should_receive(:charge_to_the_limit).
    and_return(:ok)

  assert_equal :ok,
    my_credit_card.charge_to_the_limit
end
```

Real Object with a single mocked method

Classes are Objects Too

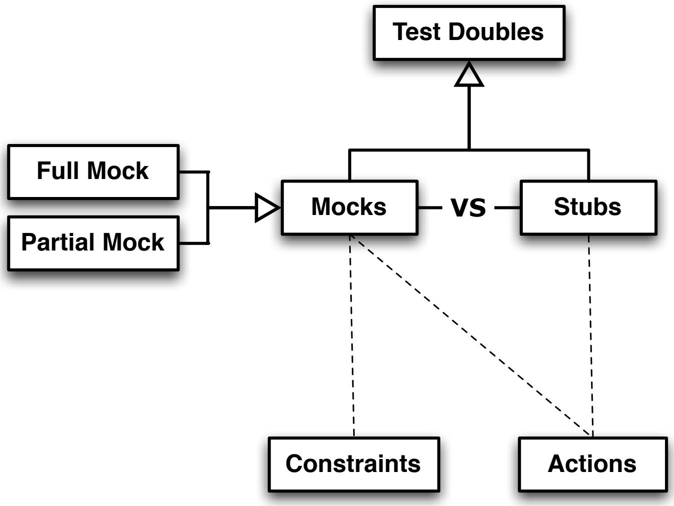
```
def test_something
  flexmock(User).should_receive(:find).
    and_return(User.new(...))

  # Code that needs to find a
  # user to manipulate
end
```

Injecting Dependencies

```
def test_value_of_a_single_stock
  fake = flexmock("quote svc")
  fake.should_receive(:quote).
    with("APPL").and_return(100).once
  port = Portfolio.new(fake)
  port.add_stock("APPL")

  actual = port.value

  assert_equal 100, actual
end
```

Injecting Dependencies

```
def test_value_of_a_single_stock
  fake = flexmock("quote svc")
  fake.should_receive(:quote).
    with("APPL").and_return(100).once
  flexmock(QuoteService).
    should_receive(:new).
    and_return(fake)
  port = Portfolio.new
  port.add_stock("APPL")

  actual = port.value

  assert_equal 100, actual
end
```



Test Doubles

Full Mock

Partial Mock

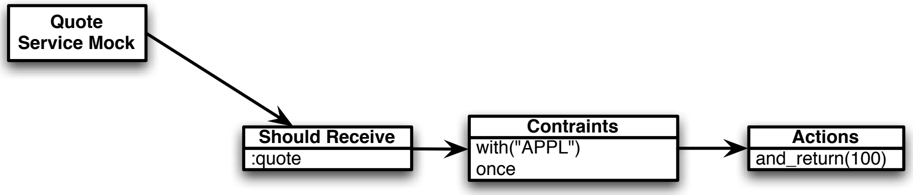
Mocks

VS

Stubs

Constraints

Actions

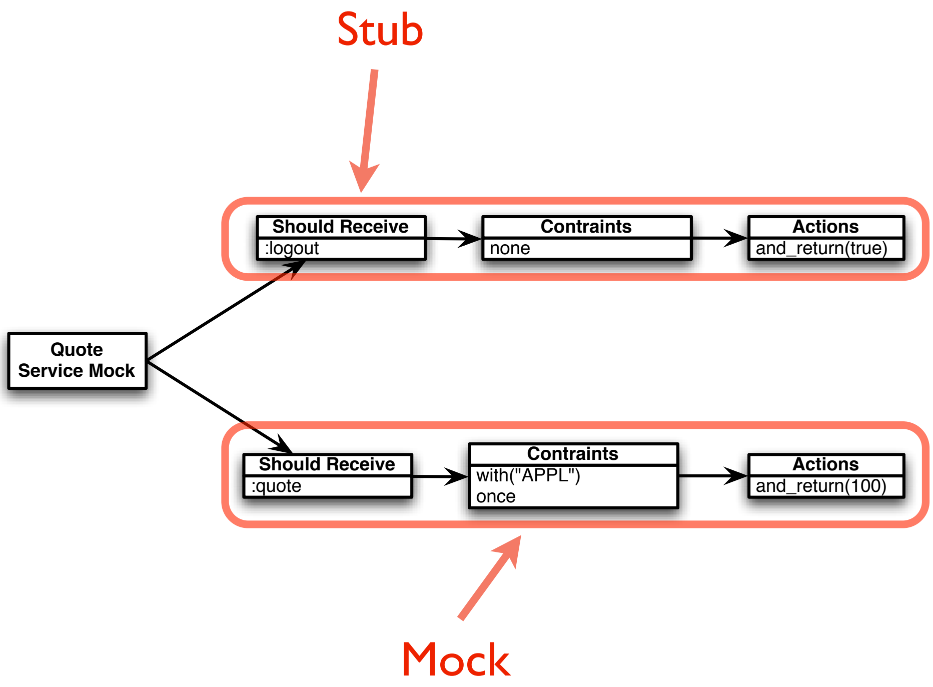


Quote Service Mock

Should Receive
:quote

Contraints
with("APPL")
once

Actions
and_return(100)



Stub

Should Receive
:logout

Contraints
none

Actions
and_return(true)

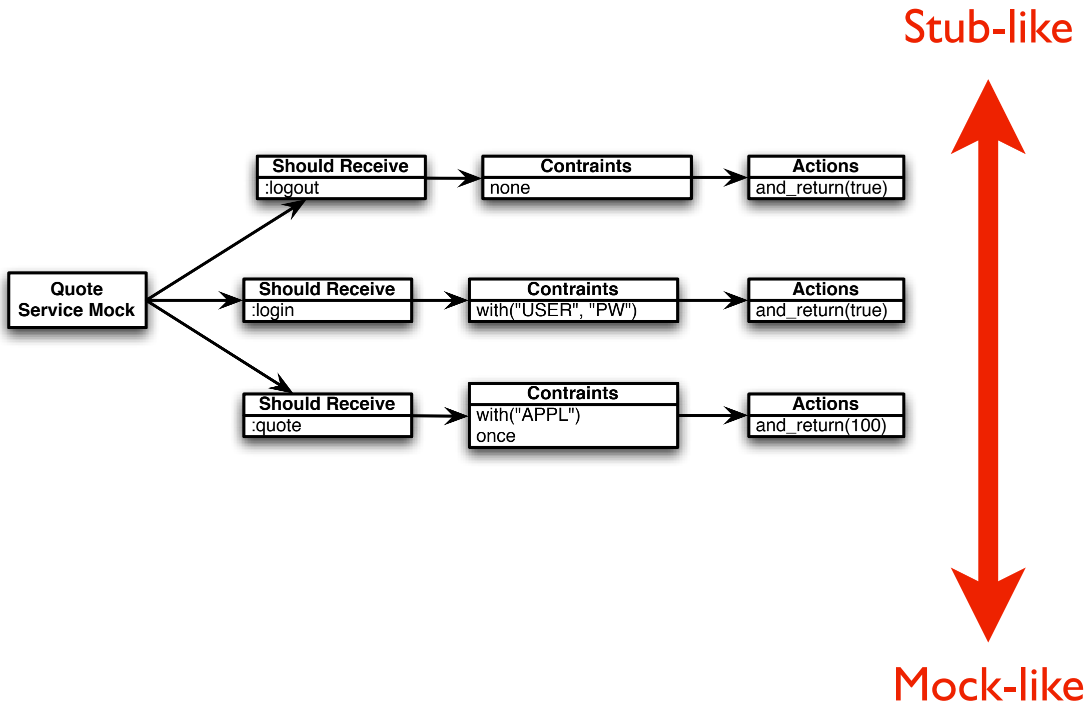
Quote
Service Mock

Should Receive
:quote

Contraints
with("APPL")
once

Actions
and_return(100)

Mock



Stub-like

Should Receive
:logout

Contraints
none

Actions
and_return(true)

Quote
Service Mock

Should Receive
:login

Contraints
with("USER", "PW")

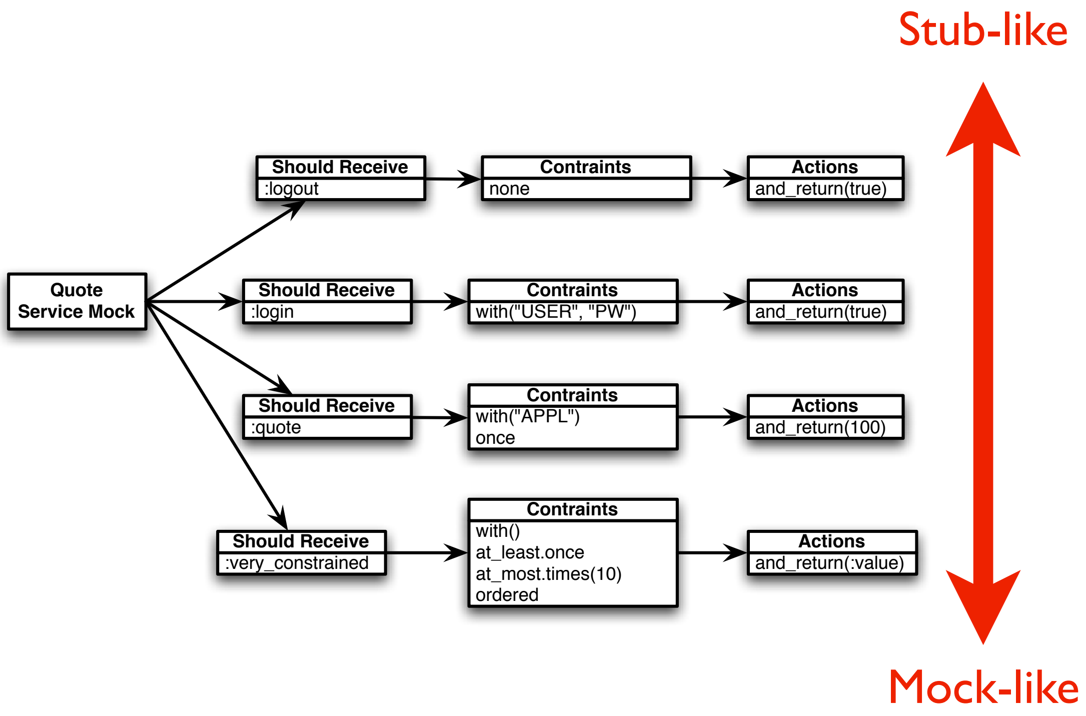
Actions
and_return(true)

Should Receive
:quote

Contraints
with("APPL")
once

Actions
and_return(100)

Mock-like



Stub-like

Quote Service Mock

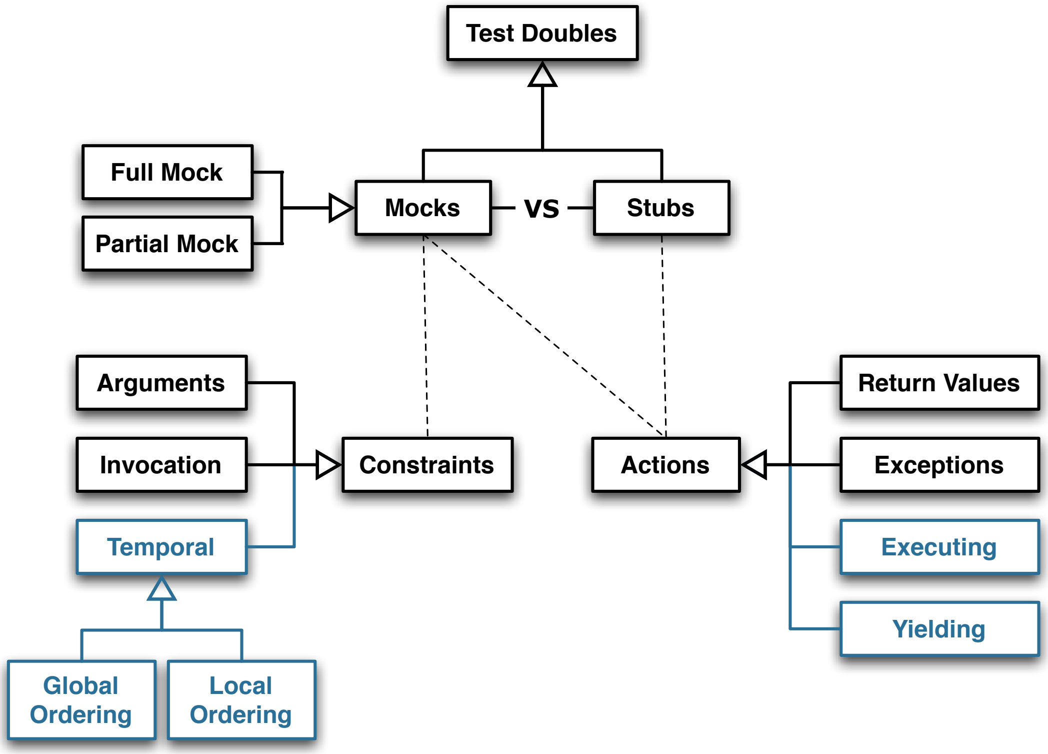
Should Receive :logout → Contraints none → Actions and_return(true)

Should Receive :login → Contraints with("USER", "PW") → Actions and_return(true)

Should Receive :quote → Contraints with("APPL") once → Actions and_return(100)

Should Receive :very_constrained → Contraints with() at_least.once at_most.times(10) ordered → Actions and_return(:value)

Mock-like



Test Doubles

Mocks VS Stubs

Full Mock

Partial Mock

Arguments

Invocation

Temporal

Global Ordering

Local Ordering

Constraints

Actions

Return Values

Exceptions

Executing

Yielding

Argument Constraints

Explicit Matches

```
mock.should_receive(:quote).with("APPL")
```

```
mock.should_receive(:logout).with()
```

```
mock.should_receive(:login).with("USER", "PW")
```

Argument Constraints

Class Matches

```
mock.should_receive(:quote).with(String)
```

```
mock.should_receive(:login).
   with("USER", String)
```

Argument Constraints

Regex Matches

```
mock.should_receive(:quote).with(/APPL|GOOG/)
```

```
mock.should_receive(:login).
   with("USER", /\S{6,10}/)
```

Argument Constraints

Anything

```
mock.should_receive(:quote).with(any)
```

```
mock.should_receive(:login).with(any, any)
```

Argument Constraints

Exact Match

```
mock.should_receive(:is_a?).with(eq(String))
```

```
mock.should_receive(:match).with(eq(/.*/))
```

Invocation Constraints

Exact Count

```
mock.should_receive(:foo).once
```

```
mock.should_receive(:foo).twice
```

```
mock.should_receive(:foo).times(10)
```

```
mock.should_receive(:foo).never
```

Actions

Return Value

```
mock.should_receive(:login).and_return(true)
```

Actions

Multiple Return Value

```
mock.should_receive(:quote).times(3).
  and_return(100, 20, 3)
```

Actions

Exceptions

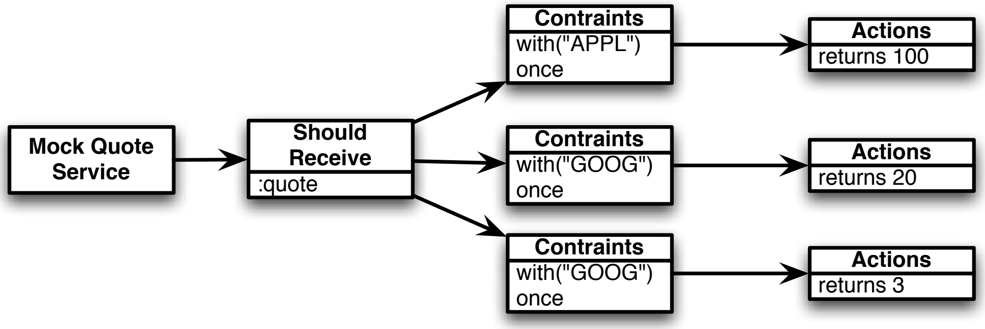
```
mock.should_receive(:login).
  and_raise(LoginError)
```

```
mock.should_receive(:login).
  and_raise(LoginError, "Invalid Login")
```

```
mock.should_receive(:login).
  and_raise(LoginError.new("Invalid Login"))
```

Method Resolution

```
mock.should_receive(:quote).once.with("APPL").
  and_return(100)

mock.should_receive(:quote).once.with("GOOG").
  and_return(20)

mock.should_receive(:quote).once.with("GOOG").
  and_return(3)
```



Mock Quote Service

Should Receive
:quote

Contraints
with("APPL")
once

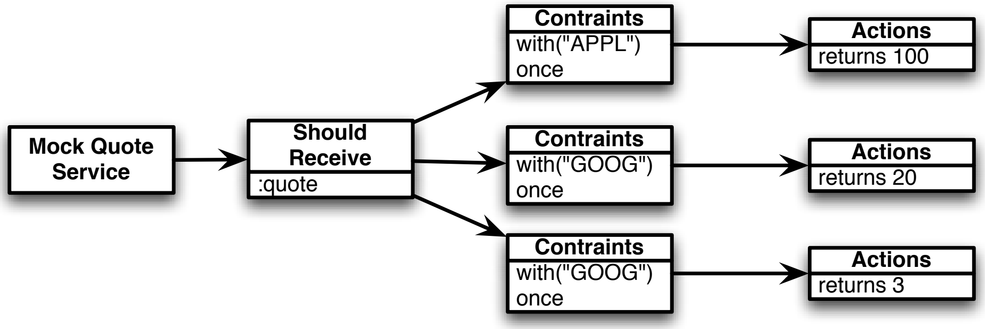
Contraints
with("GOOG")
once

Contraints
with("GOOG")
once

Actions
returns 100

Actions
returns 20

Actions
returns 3



Mock Quote Service

Should Receive
:quote

Contraints
with("APPL")
once

Contraints
with("GOOG")
once

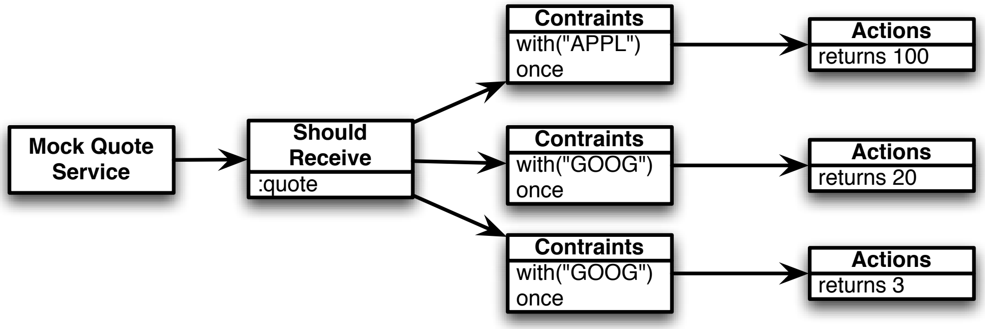
Contraints
with("GOOG")
once

Actions
returns 100

Actions
returns 20

Actions
returns 3

```
mock.quote("GOOG")    # => ?
mock.quote("GOOG")    # => ?
mock.quote("APPL")    # => ?
mock.quote("APPL")    # => ?
```



Mock Quote Service

Should Receive
:quote

Contraints
with("APPL")
once

Contraints
with("GOOG")
once

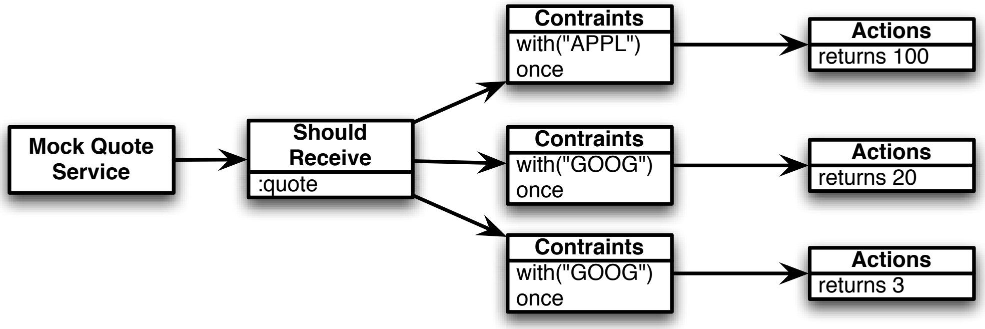
Contraints
with("GOOG")
once

Actions
returns 100

Actions
returns 20

Actions
returns 3

```
mock.quote("GOOG")    # => 20
mock.quote("GOOG")    # => ?
mock.quote("APPL")    # => ?
mock.quote("APPL")    # => ?
```



Mock Quote Service

Should Receive
:quote

Contraints
with("APPL")
once

Contraints
with("GOOG")
once

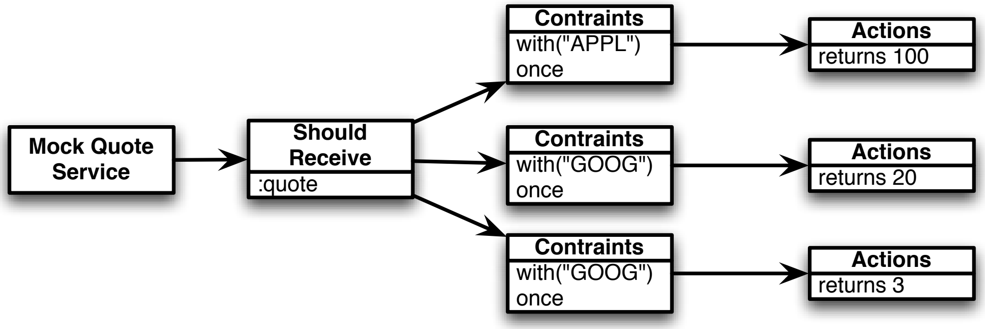
Contraints
with("GOOG")
once

Actions
returns 100

Actions
returns 20

Actions
returns 3

```
mock.quote("GOOG")    # => 20
mock.quote("GOOG")    # => 3
mock.quote("APPL")    # => ?
mock.quote("APPL")    # => ?
```



Mock Quote Service

Should Receive
:quote

Contraints
with("APPL")
once

Contraints
with("GOOG")
once

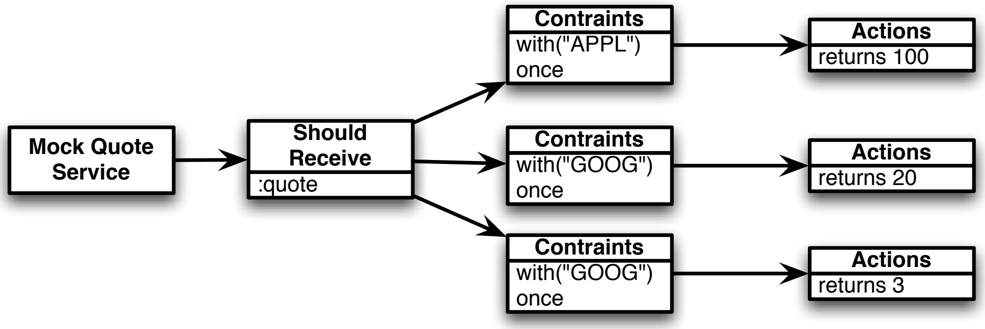
Contraints
with("GOOG")
once

Actions
returns 100

Actions
returns 20

Actions
returns 3

```
mock.quote("GOOG")     # => 20
mock.quote("GOOG")     # => 3
mock.quote("APPL")     # => 100
mock.quote("APPL")     # => ?
```



Mock Quote Service

Should Receive
:quote

Contraints
with("APPL")
once

Contraints
with("GOOG")
once

Contraints
with("GOOG")
once

Actions
returns 100

Actions
returns 20

Actions
returns 3

```
mock.quote("GOOG")     # => 20
mock.quote("GOOG")     # => 3
mock.quote("APPL")     # => 100
mock.quote("APPL")     # => ERROR
```

Stubbing Shortcut

```
mock = flexmock("a mock")
mock.should_receive(:foo).and_return(10)
mock.should_receive(:bar).and_return("value")
```

Stubbing Shortcut

```
mock = flexmock("a mock")
mock.should_receive(:foo).and_return(10)
mock.should_receive(:bar).and_return("value")
```

↓

```
mock = flexmock("a mock")
mock.should_receive(:foo => 10, :bar => "value")
```

Stubbing Shortcut

```
mock = flexmock("a mock")
mock.should_receive(:foo).and_return(10)
mock.should_receive(:bar).and_return("value")
```

```
mock = flexmock("a mock",
          :foo => 10, :bar => "value")
```

When to Mock

Not

When‸to Mock

Mock Pros & Cons

- Pros
- Cons

Mock Pros & Cons

- Pros
  - Mocks disconnect you from the real object
- Cons

Mock Pros & Cons

- Pros
  - Mocks disconnect you from the real object
- Cons
  - Mocks disconnect you from the real object

What Breaks?

```
def test_value_is_sum_of_stock_values
  fake = flexmock("quote_svc")
  fake.should_receive(:quote).
    with("APPL").and_return(100).once
  port = Portfolio.new(fake)
  port.add_stock("APPL")

  actual = port.value

  assert_equal 100, actual
end
```

Fantasy Tests

- Fail when the software is good
- Pass when the software is bad

Use Mocks When ...

- The software under tests uses an external resource (web service, database, random data)
- When the **interaction** between software under test and the supporting object is important
- ~~When it is difficult to construct the supporting object.~~

Tips

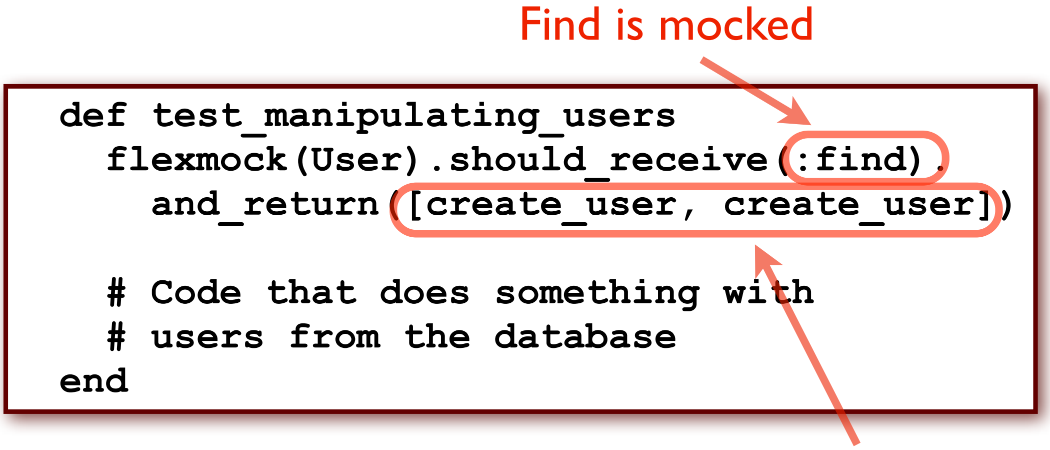
Use Factory Patterns for Easy Object Creation

(in-memory creation as well as database fixtures)

Tips

Take Advantage of Partial Mocks

Mock Finders



Find is mocked

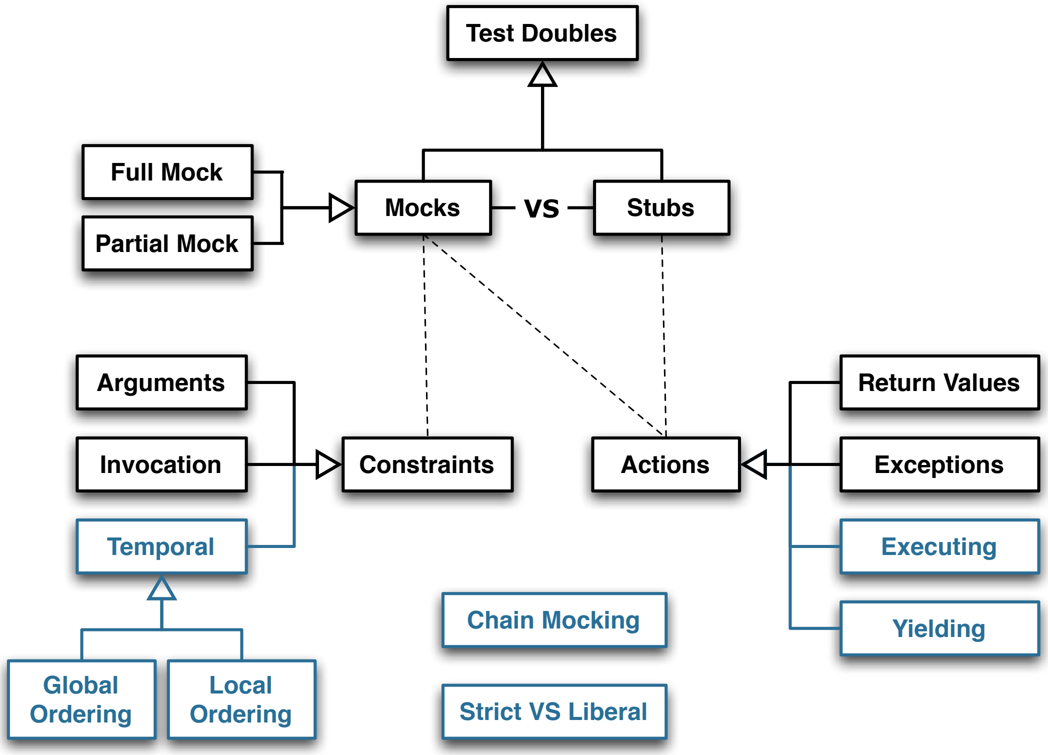
```
def test_manipulating_users
  flexmock(User).should_receive(:find).
    and_return([create_user, create_user])

  # Code that does something with
  # users from the database
end
```

Real User Objects

NOTE: We are *not* testing the find!

Tips

Don't Forgot Integration Testing



Test Driven Development with Rails

Lab 5
Mocking

Lab 5: Mocking

- Complete the Quote Service Example
  - Login must be called
  - Logout must be called
  - Quote must be called multiple times, once for each tracked quote
  - Value returned from quote must be the sum of all the single stock values

Lab 5: Mocking

- Complete the Quote Service Example
  - Login must be called
  - Logout must be called
  - Quote must be called multiple times, once for each tracked quote
  - Value returned from quote must be the sum of all the single stock values

Extra Credit

  - Quote returns nil if login fails
  - Quote returns good value if logout fails

Test Driven Development with Rails

Rails Testing

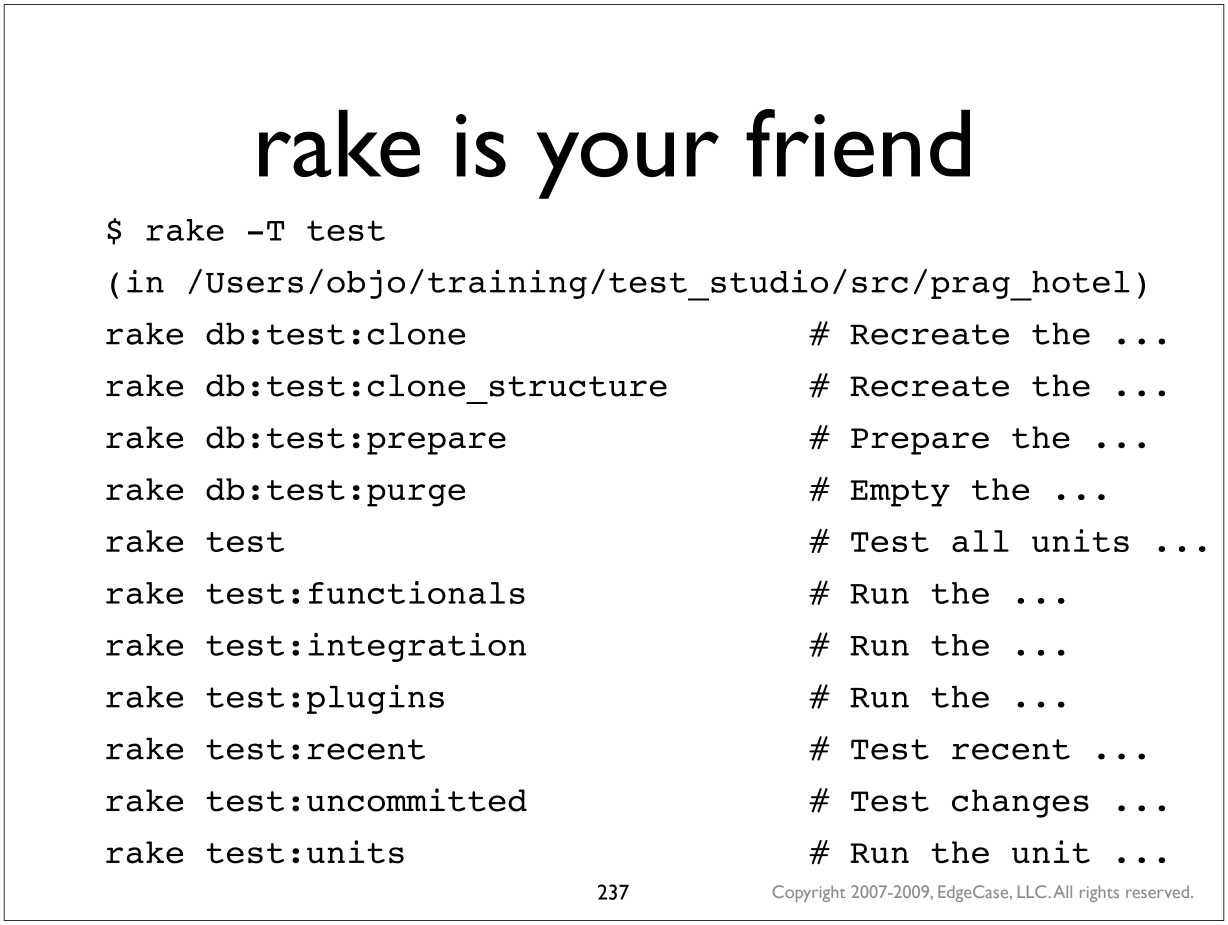
Environments

- Production
- Development
- Testing

RAILS_ENV = test

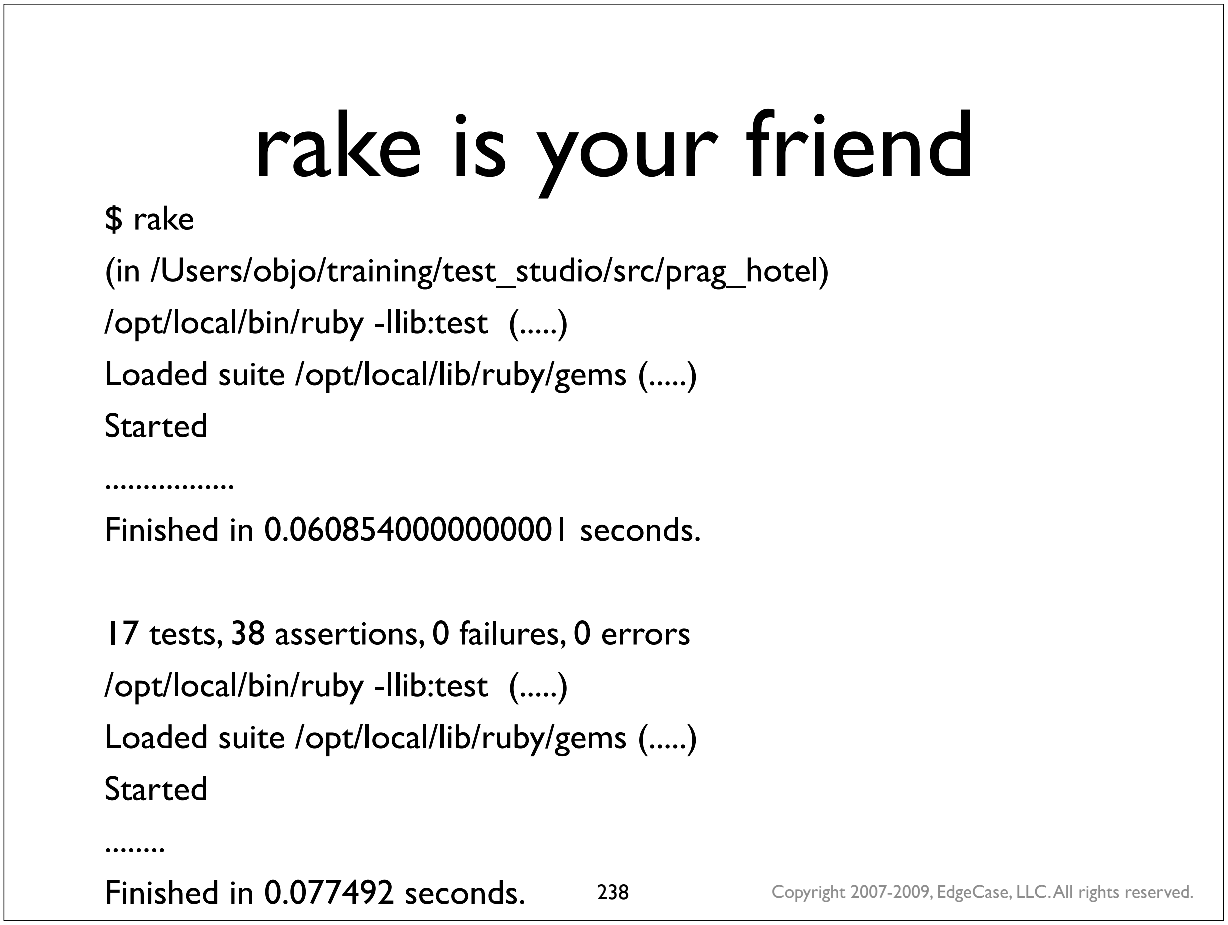
- Database is highly volatile
- Errors are thrown explicitly
- Action Mailer does not send out
- Specialized environment

```
config/environments/test.rb
```



rake is your friend

```
$ rake -T test
(in /Users/objo/training/test_studio/src/prag_hotel)
rake db:test:clone                  # Recreate the ...
rake db:test:clone_structure        # Recreate the ...
rake db:test:prepare                # Prepare the ...
rake db:test:purge                  # Empty the ...
rake test                           # Test all units ...
rake test:functionals               # Run the ...
rake test:integration               # Run the ...
rake test:plugins                   # Run the ...
rake test:recent                    # Test recent ...
rake test:uncommitted               # Test changes ...
rake test:units                     # Run the unit ...
```



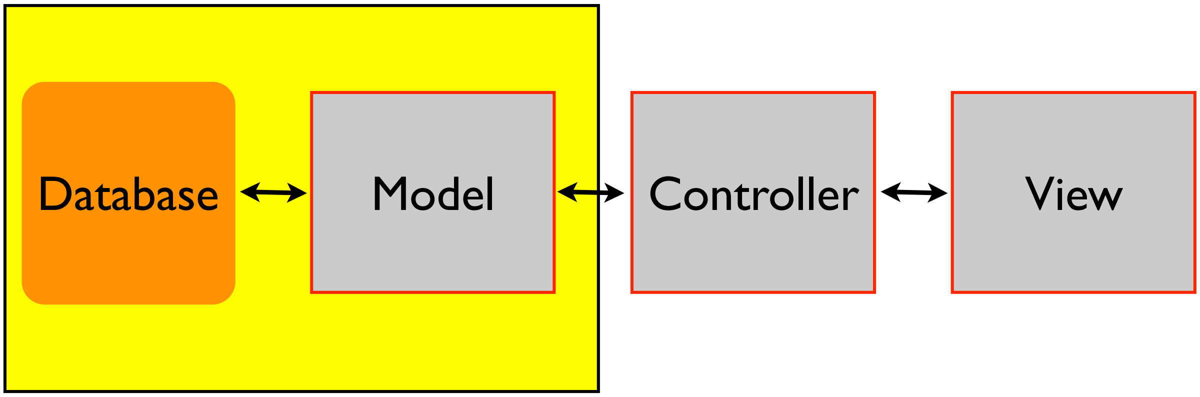
rake is your friend

```
$ rake
(in /Users/objo/training/test_studio/src/prag_hotel)
/opt/local/bin/ruby -Ilib:test  (.....)
Loaded suite /opt/local/lib/ruby/gems (.....)
Started
.................
Finished in 0.0608540000000001 seconds.

17 tests, 38 assertions, 0 failures, 0 errors
/opt/local/bin/ruby -Ilib:test  (.....)
Loaded suite /opt/local/lib/ruby/gems (.....)
Started
........
Finished in 0.077492 seconds.
```

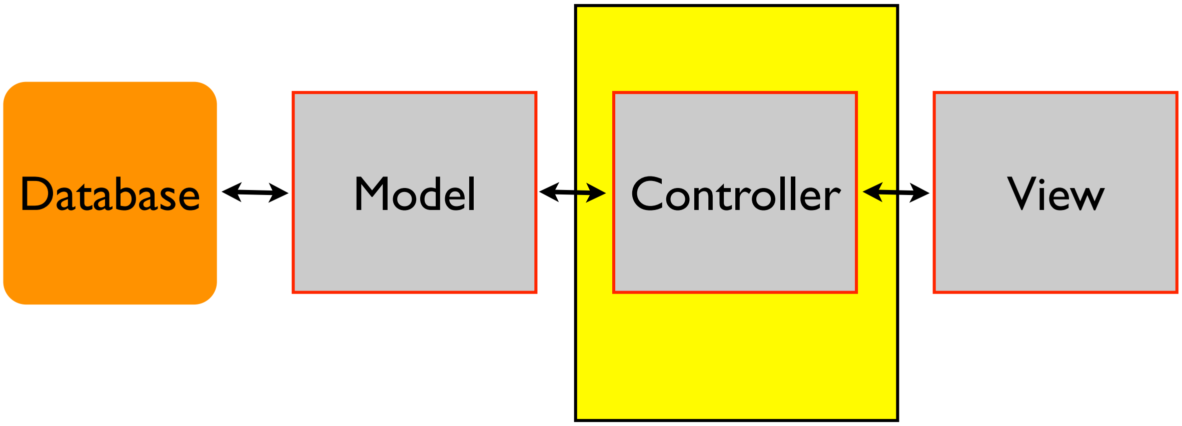
Rails Test Classification

- Unit
- Functional
- Integration

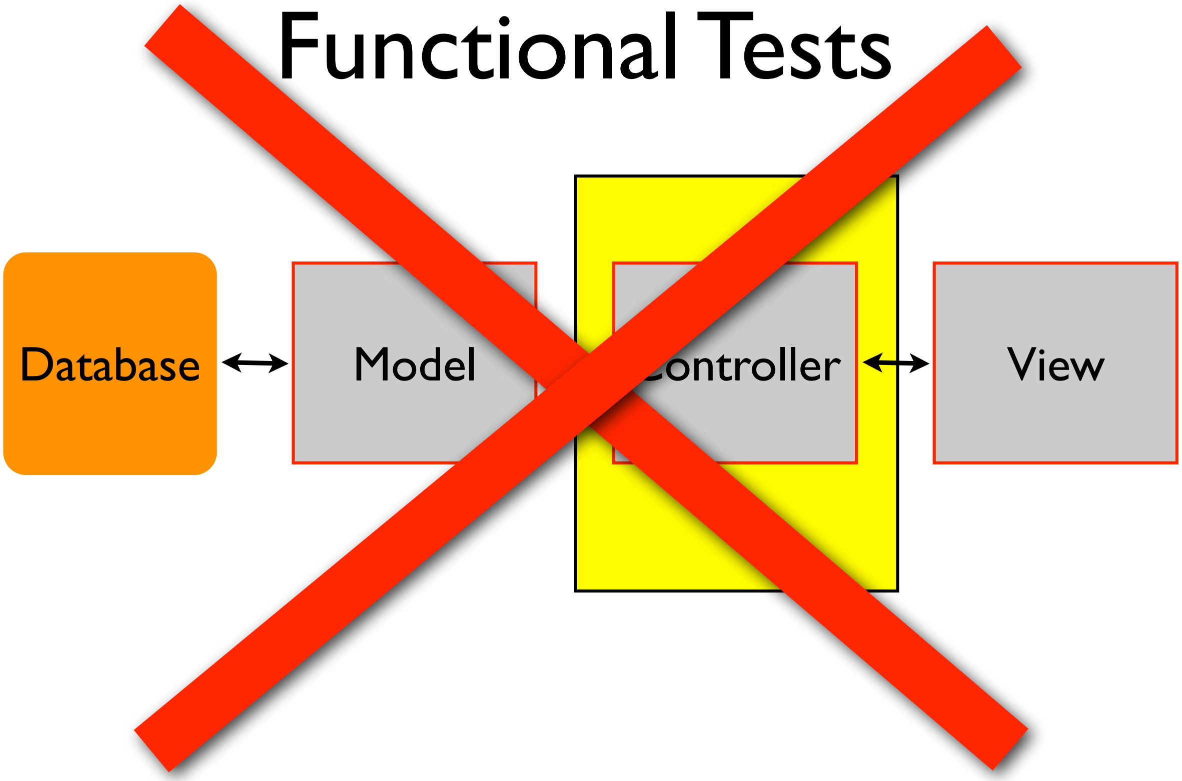
Unit Tests



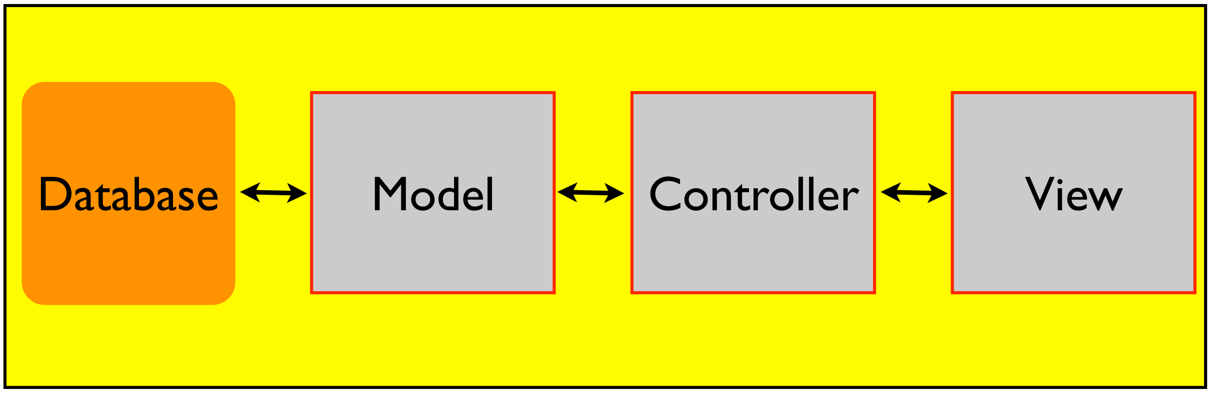
Functional Tests



Functional Tests

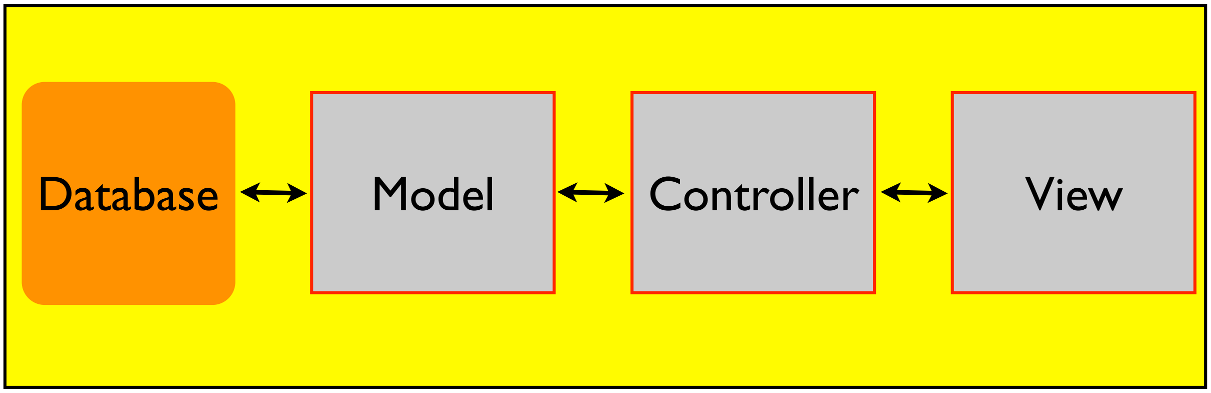


Functional Tests



Database ↔ Model ↔ Controller ↔ View

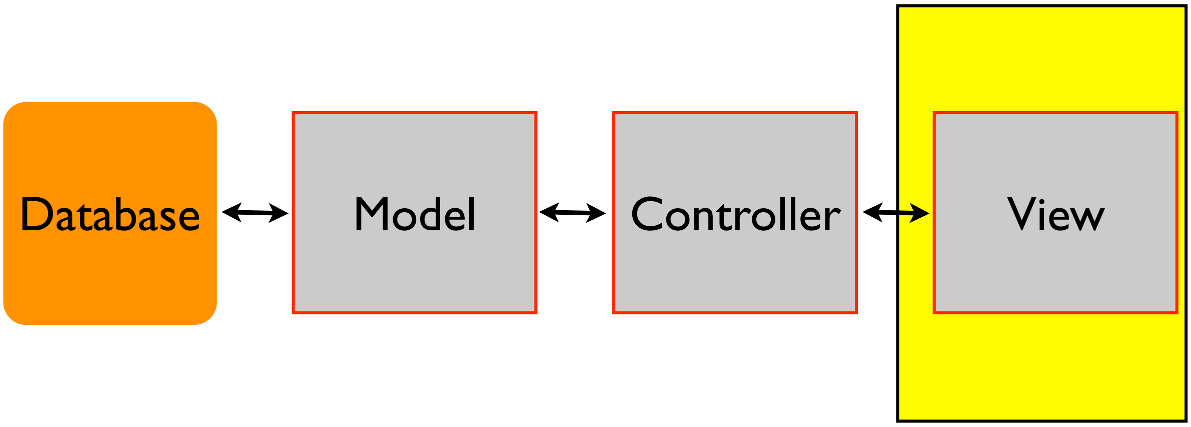
Functional Tests



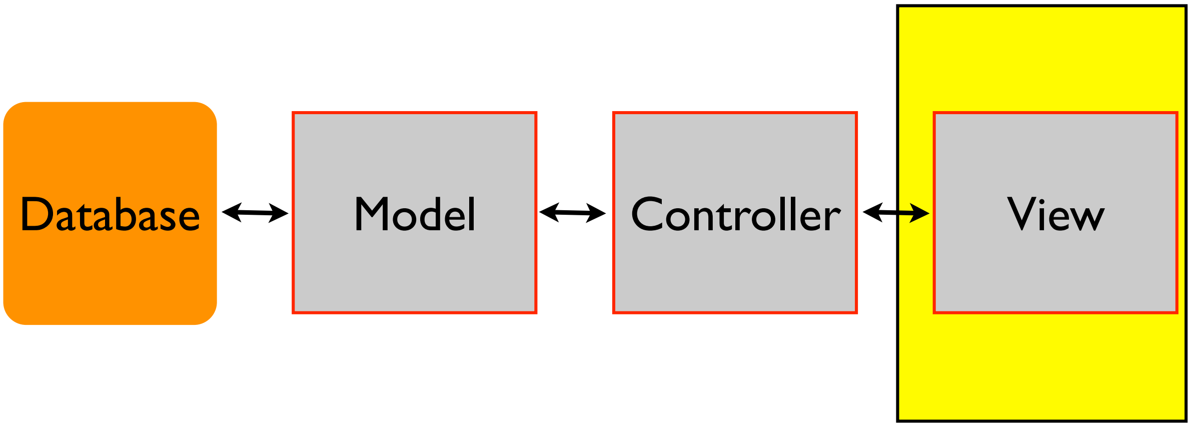
Database ↔ Model ↔ Controller ↔ View

Problem !

View Tests ?

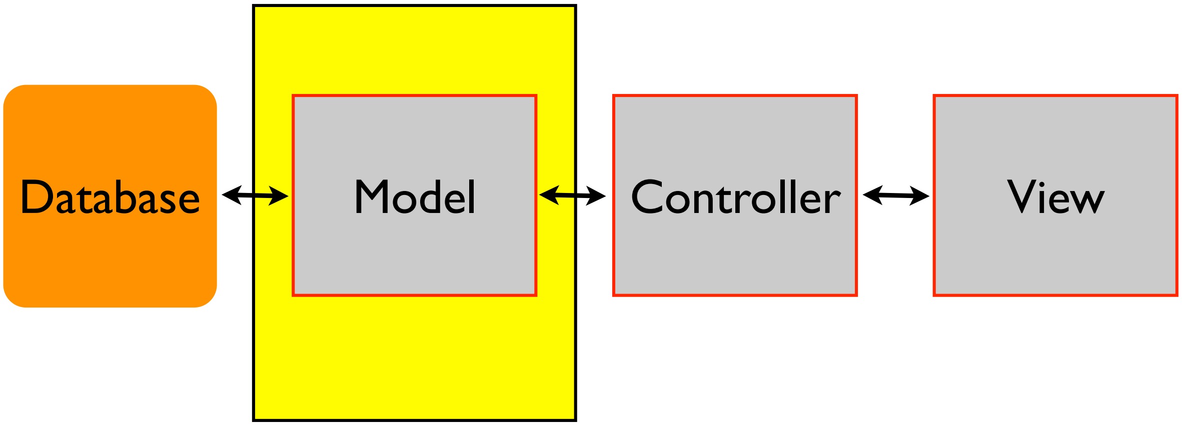


View Tests ?



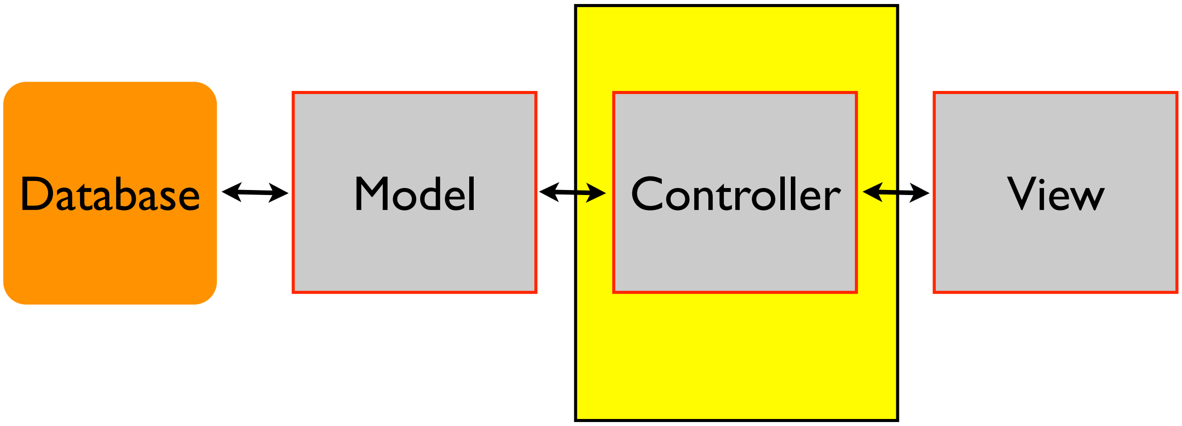
nope

Unit Testing



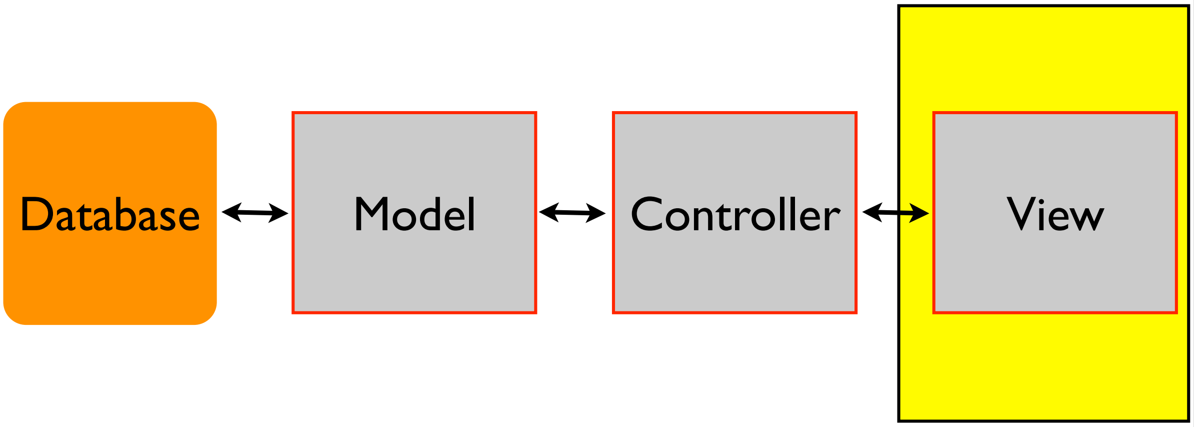
How do we get here ...

Unit Testing



... and here ...

Unit Testing

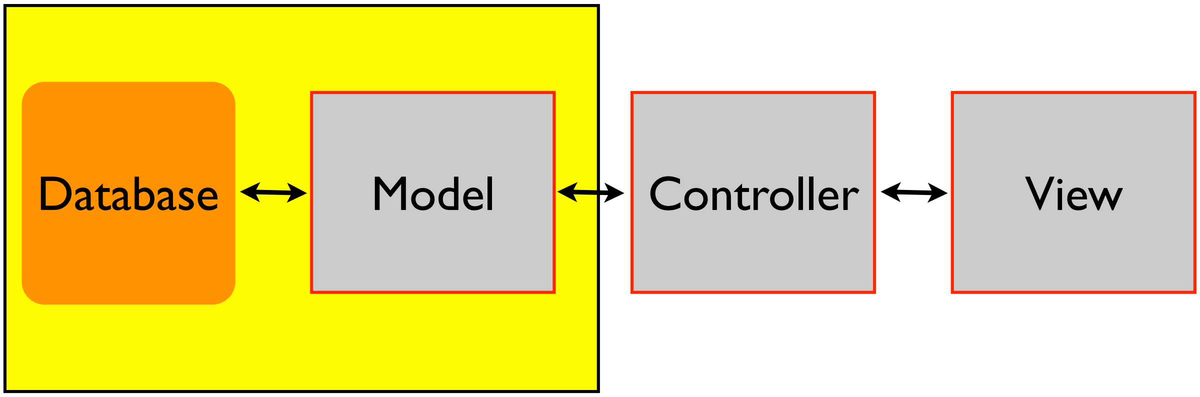


... and here ?

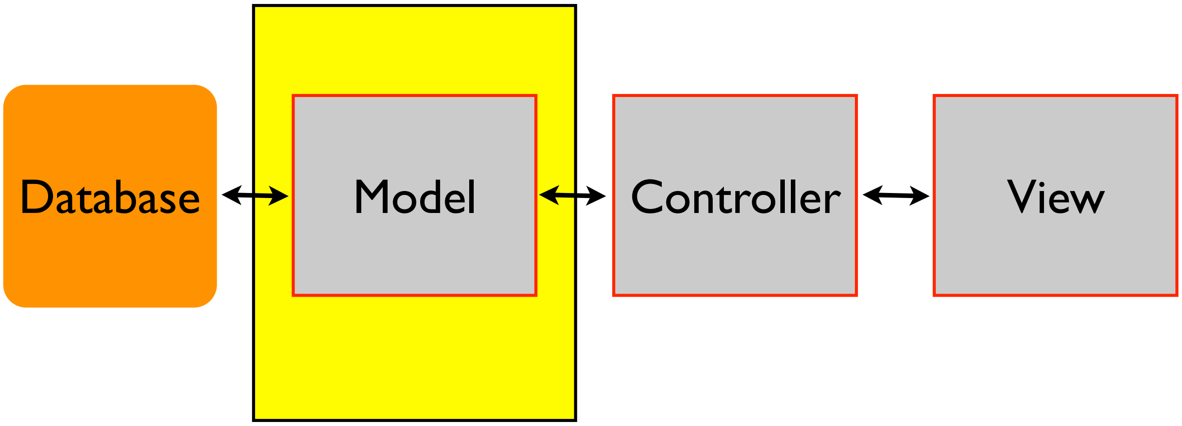
Test Driven Development with Rails

Testing Models

Unit Tests

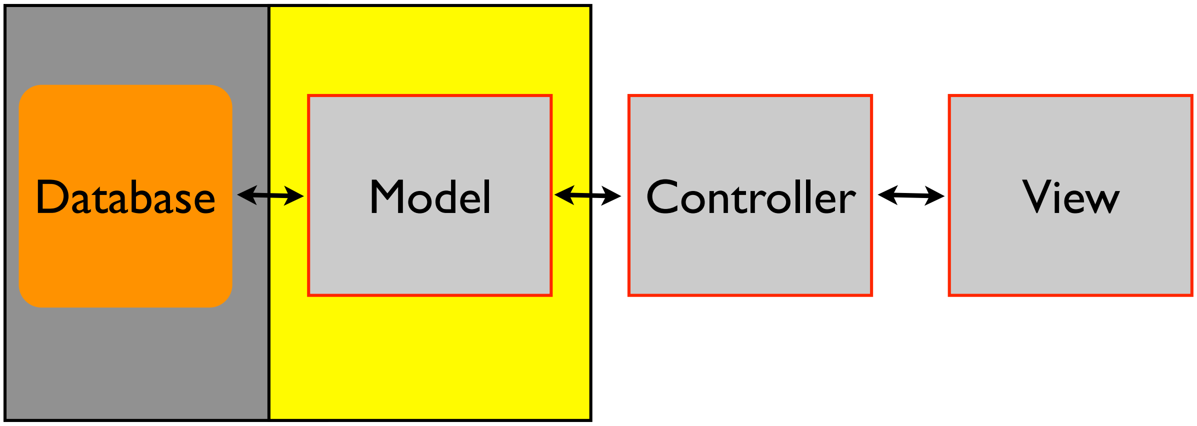


Model Tests



How do we get here ...

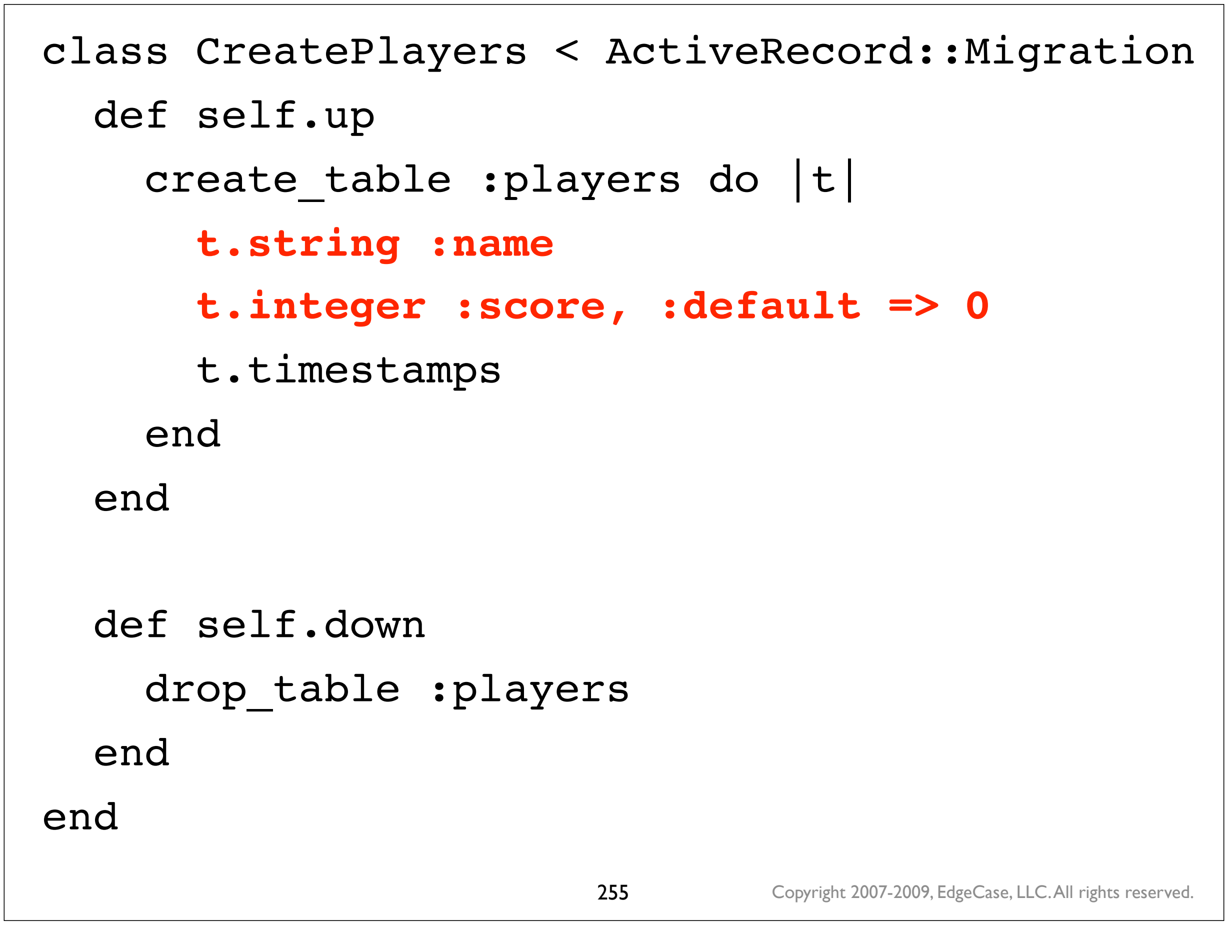
Model Tests



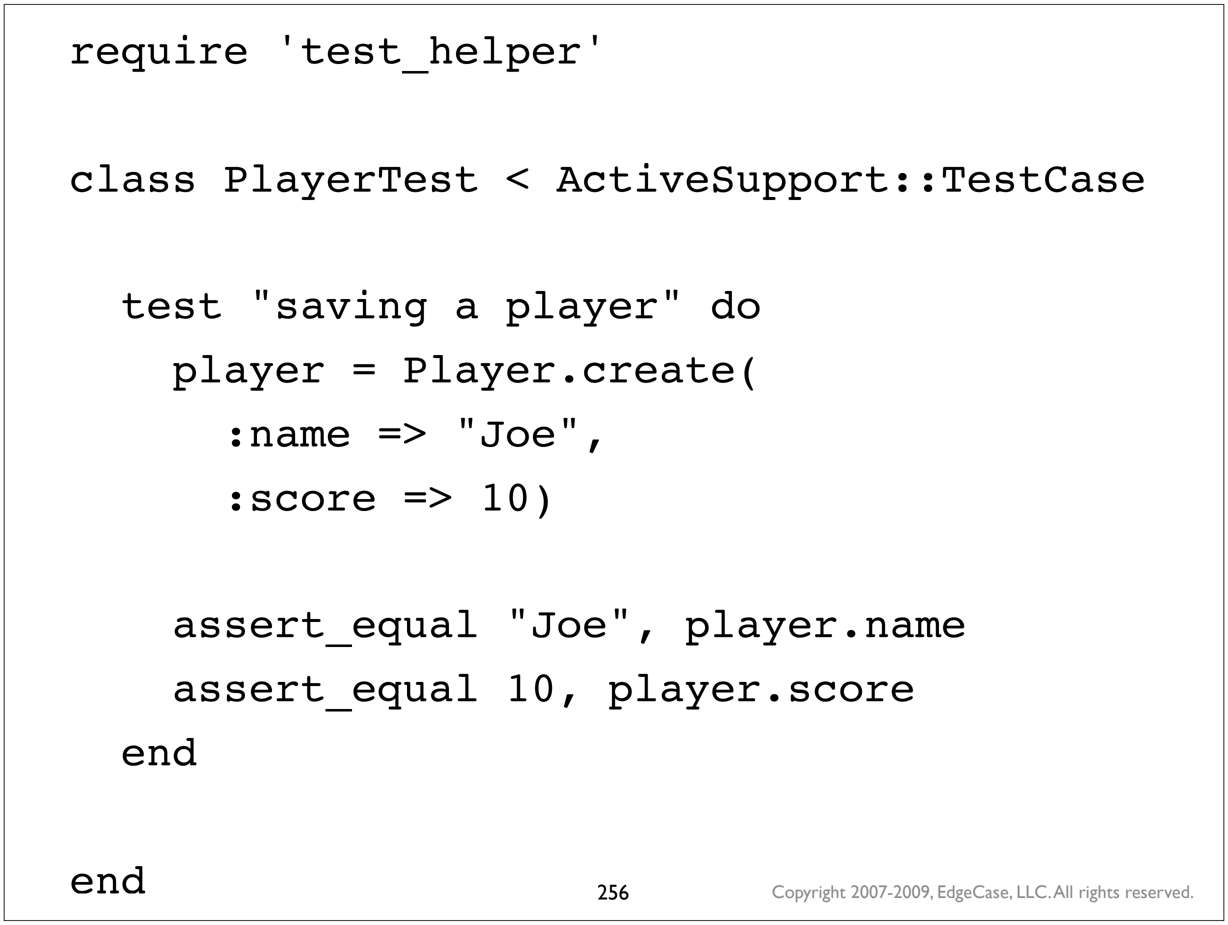
It’s more of a grey area

Player Model

```
$ script/generate model Player
```



```
class CreatePlayers < ActiveRecord::Migration
  def self.up
    create_table :players do |t|
      t.string :name
      t.integer :score, :default => 0
      t.timestamps
    end
  end

  def self.down
    drop_table :players
  end
end
```



```
require 'test_helper'

class PlayerTest < ActiveSupport::TestCase

  test "saving a player" do
    player = Player.create(
      :name => "Joe",
      :score => 10)

    assert_equal "Joe", player.name
    assert_equal 10, player.score
  end

end
```

What’s wrong here?

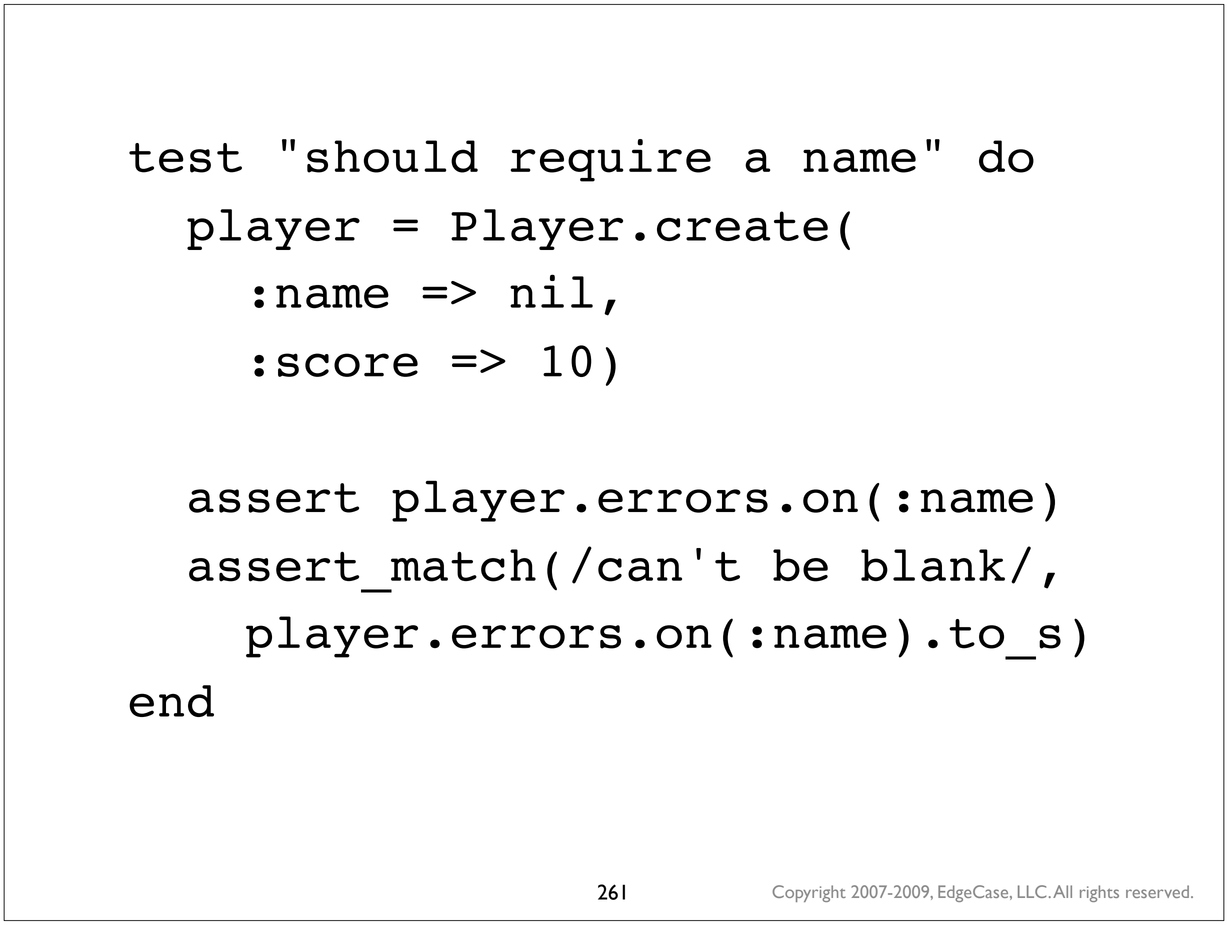
```
test "saving a player" do
  player = Player.create(
    :name => "Joe",
    :score => 10)

  assert_equal "Joe", player.name
  assert_equal 10, player.score
end
```

Testing the framework

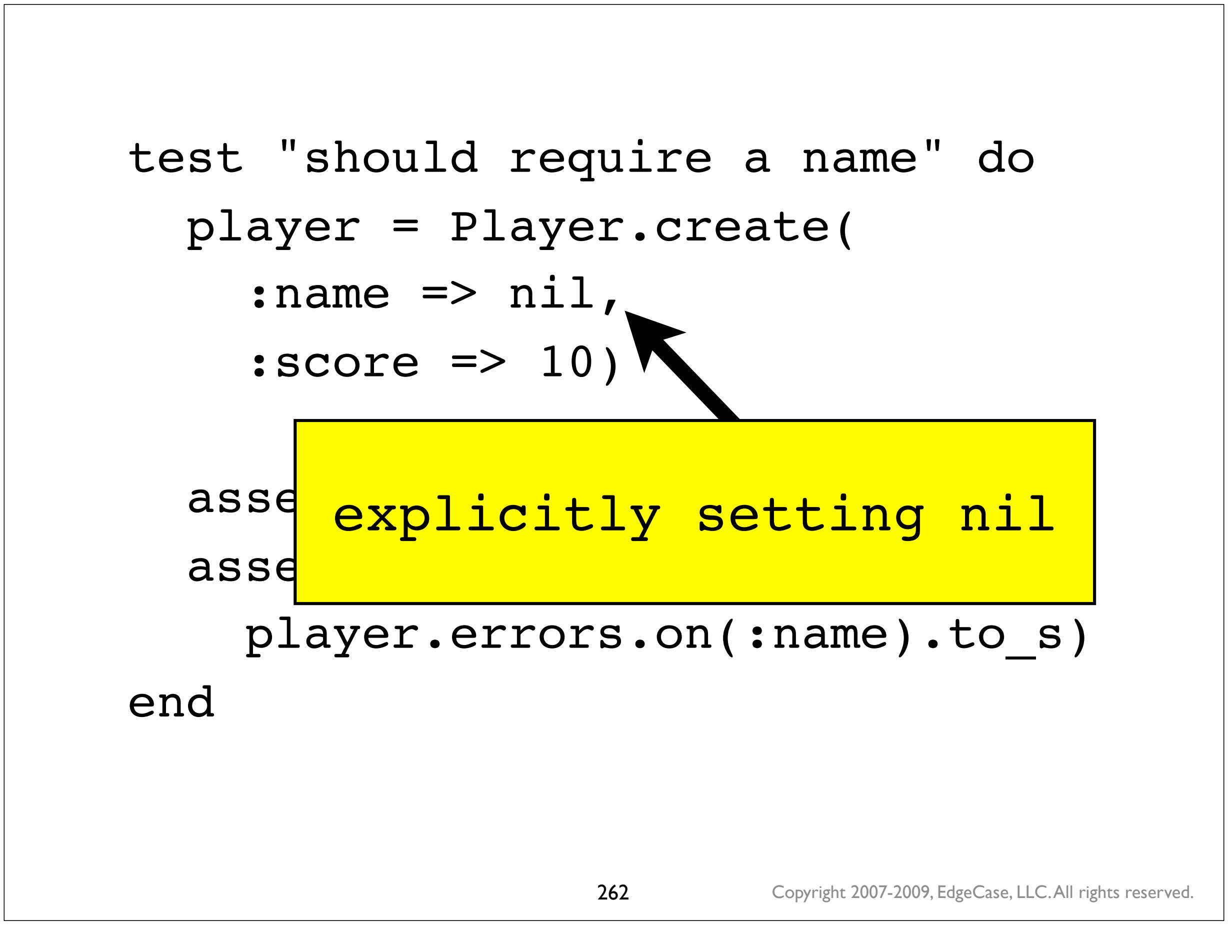
```
test "saving a player" do
  player = Player.create(
    :name => "Joe"
    :score => 10

  assert_equal "Joe", player.name
  assert_equal 10, player.score
end
```

test the behavior

validates_presence_of(:name)



```
test "should require a name" do
  player = Player.create(
    :name => nil,
    :score => 10)

  assert player.errors.on(:name)
  assert_match(/can't be blank/,
    player.errors.on(:name).to_s)
end
```



```
test "should require a name" do
  player = Player.create(
    :name => nil,
    :score => 10)

  asse
  asse
    player.errors.on(:name).to_s)
end
```

explicitly setting nil

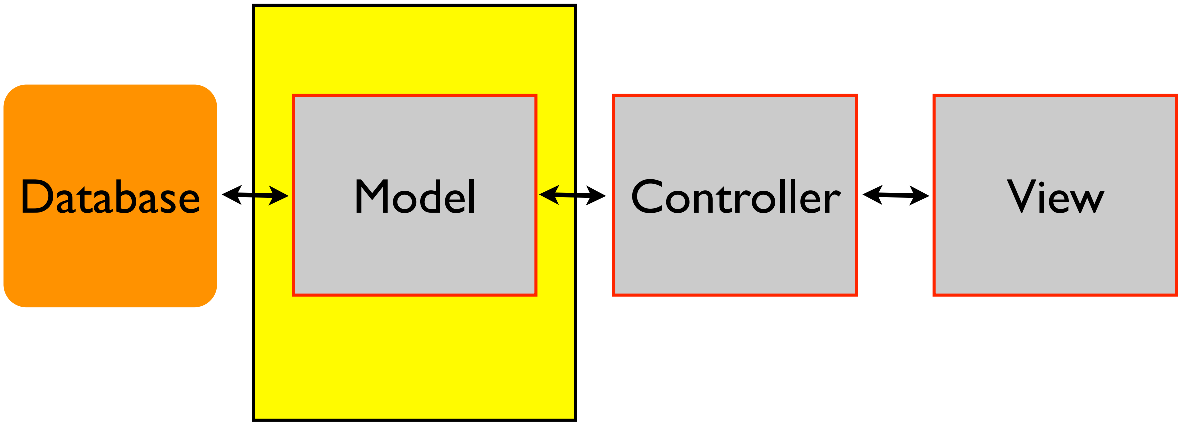
```
test "should require a name" do
  player = Player.create(
    :name => nil,
    :score => 10)

  assert player.errors.on(:name)
  assert_match(/can't be blank/,
                            e).to_s)
end
```

ensure the error
is on name

```
test "should require a name" do
  pla
    :
    :          ,

  assert player.errors.on(:name)
  assert_match(/can't be blank/,
    player.errors.on(:name).to_s)
end
```

ensure the message
is correct

Model Tests

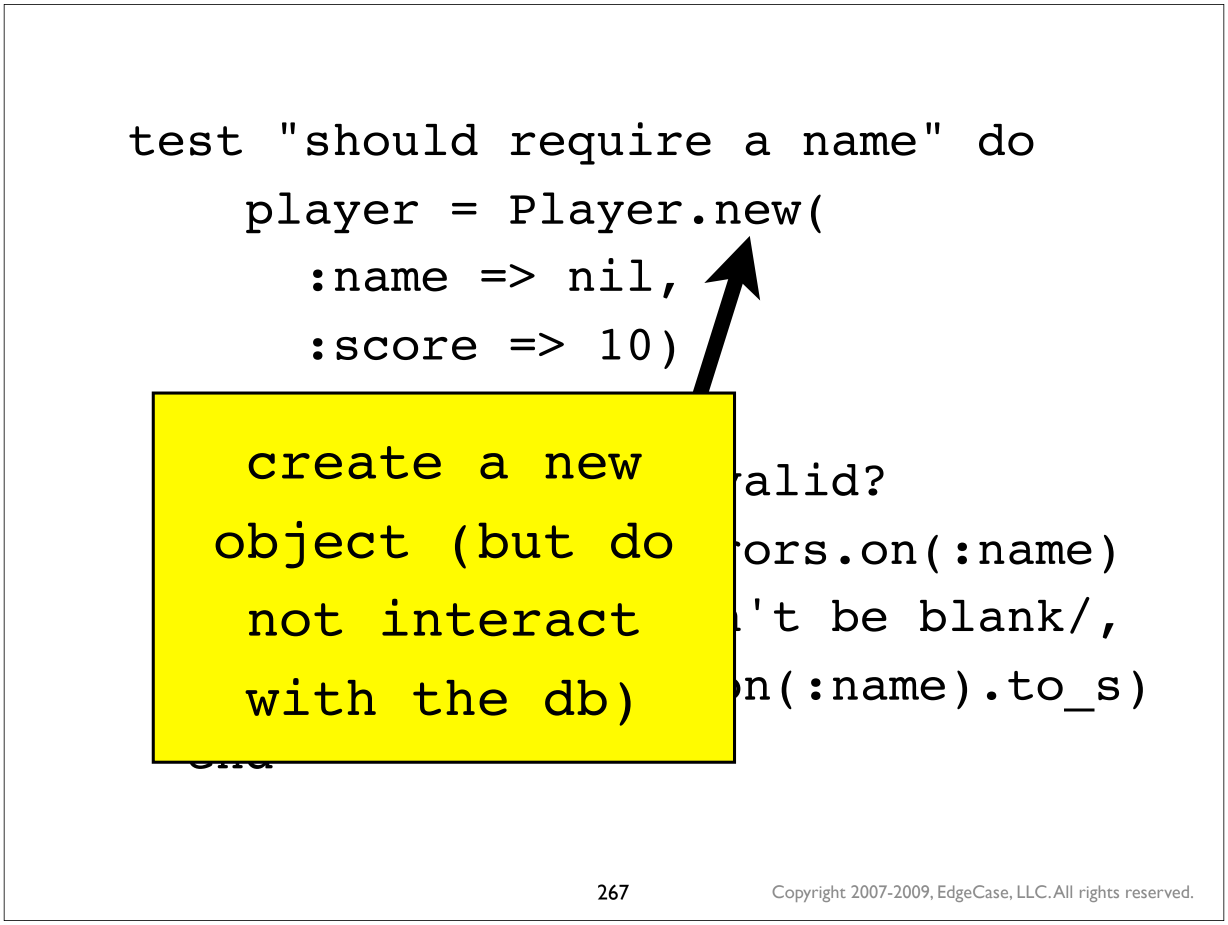


can we get closer?

```
test "should require a name" do
  player = Player.create(
    :name => nil,
    :score => 10)
```

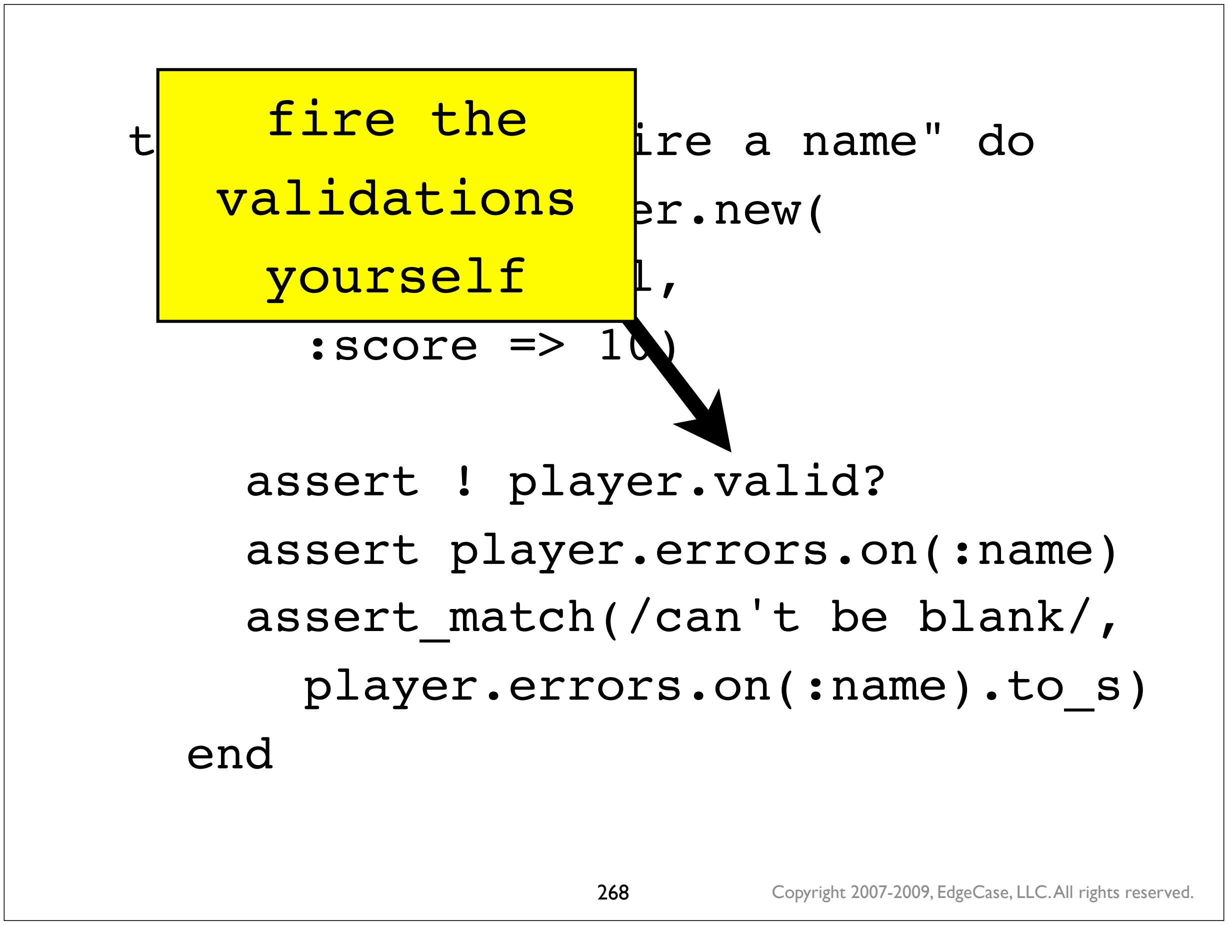
this fires the validation, but creates a record in the database

```
.on(:name)
be blank/,
name).to_s)
```



```
test "should require a name" do
    player = Player.new(
      :name => nil,
      :score => 10)

                          alid?
                          ors.on(:name)
                          't be blank/,
                          n(:name).to_s)
```

create a new object (but do not interact with the db)



fire the validations yourself

```
t                       ire a name" do
                        er.new(
                        l,
      :score => 10)

    assert ! player.valid?
    assert player.errors.on(:name)
    assert_match(/can't be blank/,
      player.errors.on(:name).to_s)
  end
```

```
test "should require a name" do
    player = Player.new(
      :name => nil
      :score => 10

    assert ! player.valid?
    assert player.errors.on(:name)
    assert_match(/can't be blank/,
      player.errors.on(:name).to_s)
  end
```

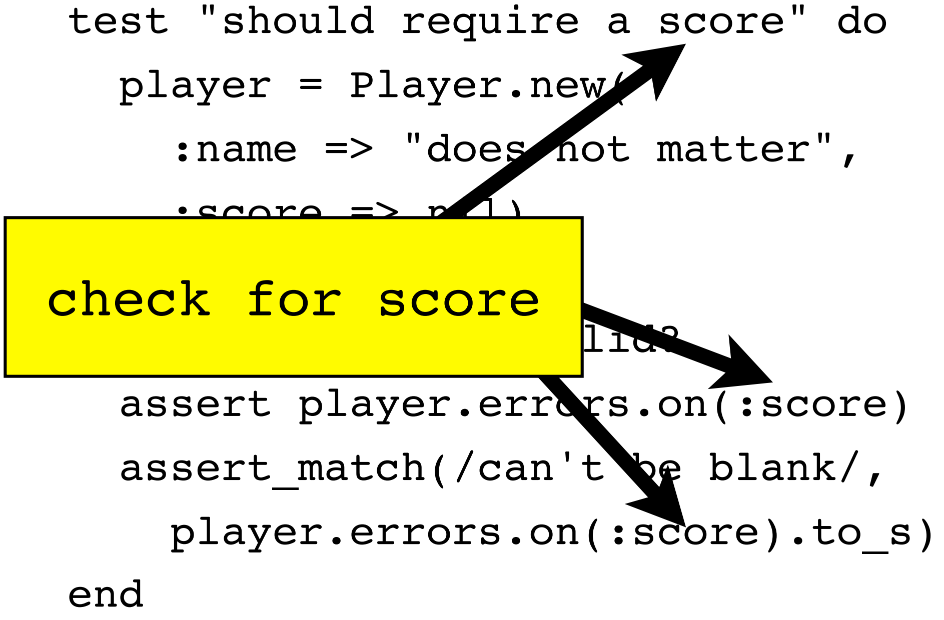
sanity check

validates_presence_of(:score)

```
test "should require a score" do
  player = Player.new(
    :name => "does not matter",
    :score => nil)

  assert ! player.valid?
  assert player.errors.on(:score)
  assert_match(/can't be blank/,
    player.errors.on(:score).to_s)
end
```

```
test "should require a score" do
  player = Player.new(
    :name => "does not matter",
    :score => nil)

  assert ! player.valid?
  assert player.errors.on(:score)
  as
end
```

document the intent

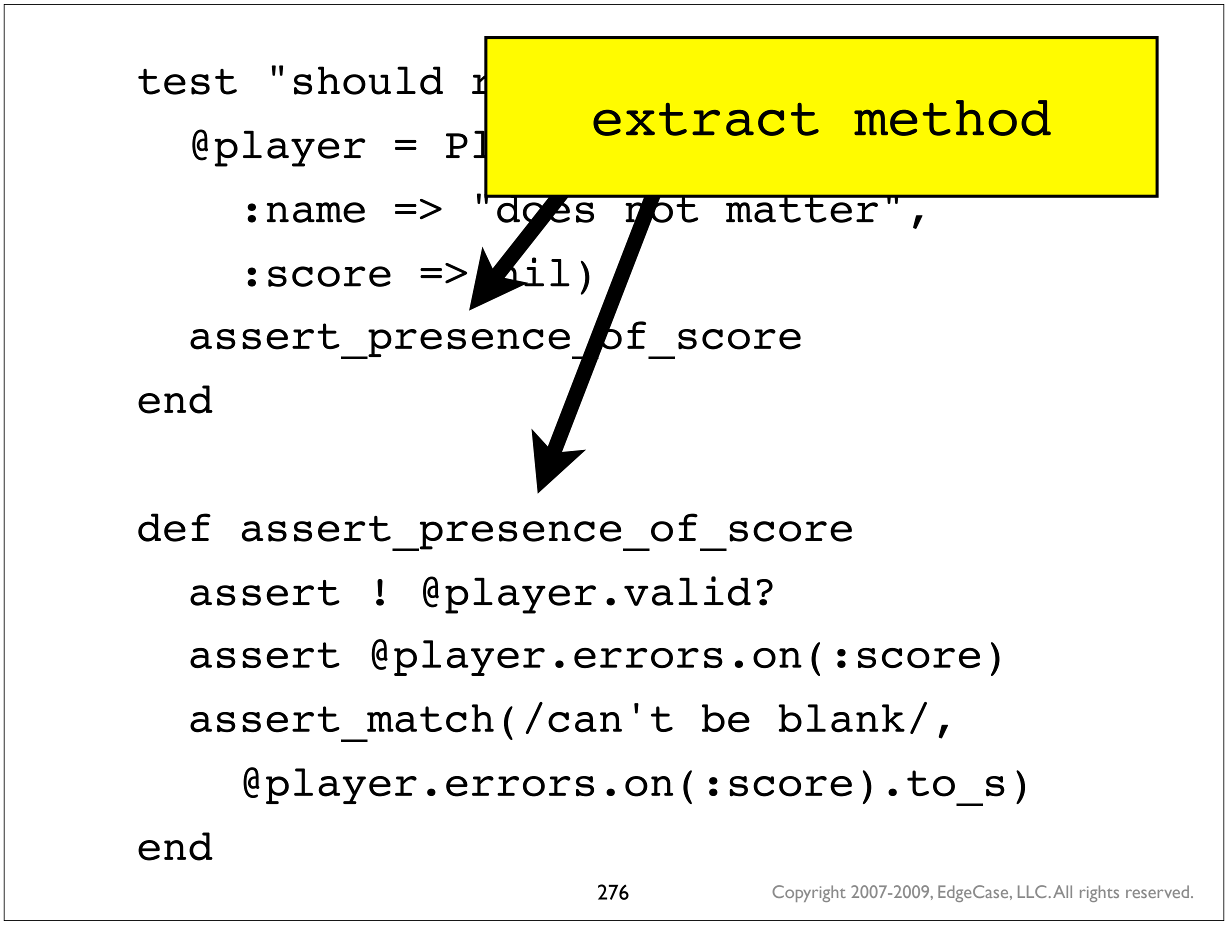


```
test "should require a score" do
  player = Player.new(
    :name => "does not matter",
    :score => nil)
                    lid?
  assert player.errors.on(:score)
  assert_match(/can't be blank/,
    player.errors.on(:score).to_s)
end
```

check for score

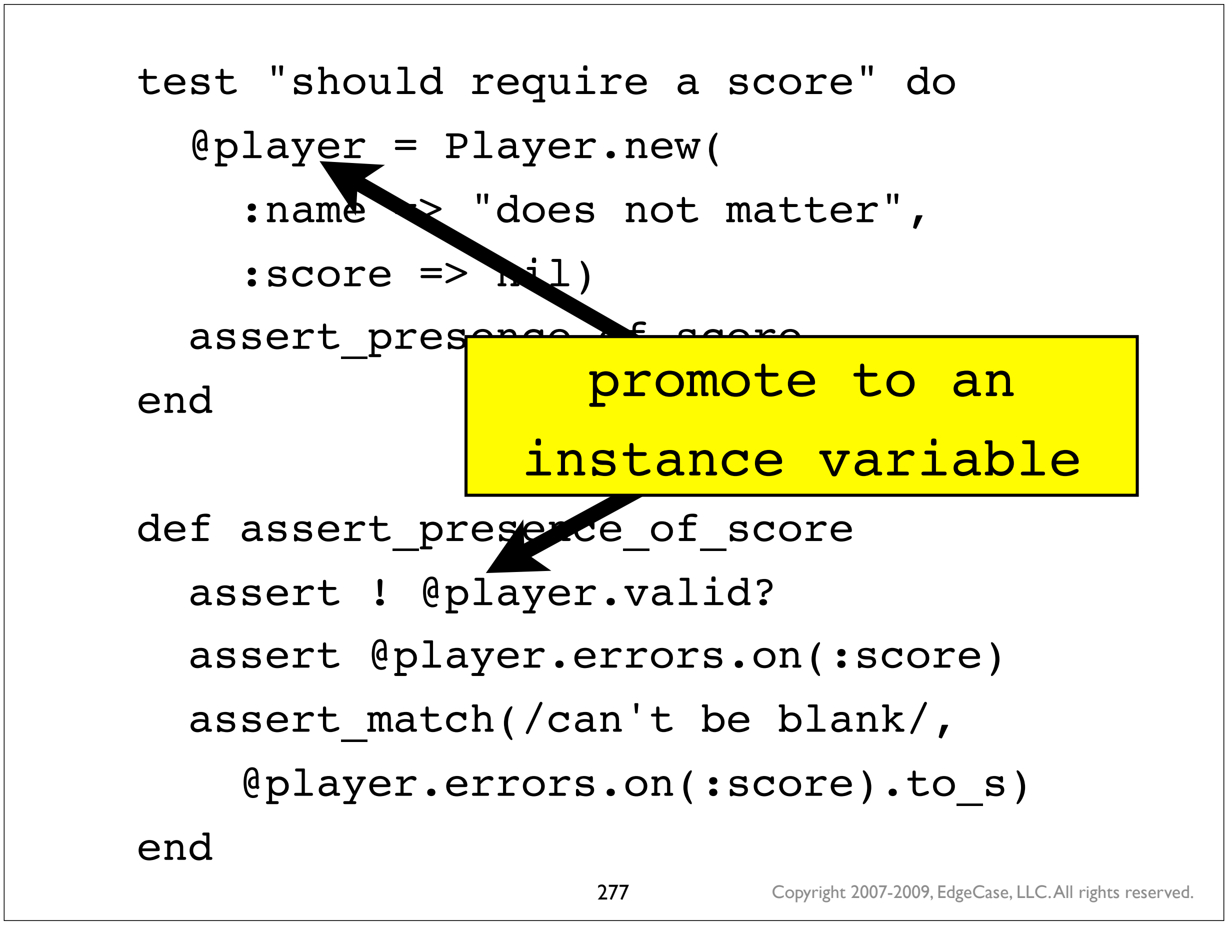
refactor:
extract method

```ruby
test "should require a score" do
  @player = Player.new(
    :name => "does not matter",
    :score => nil)
  assert_presence_of_score
end

def assert_presence_of_score
  assert ! @player.valid?
  assert @player.errors.on(:score)
  assert_match(/can't be blank/,
    @player.errors.on(:score).to_s)
end
```



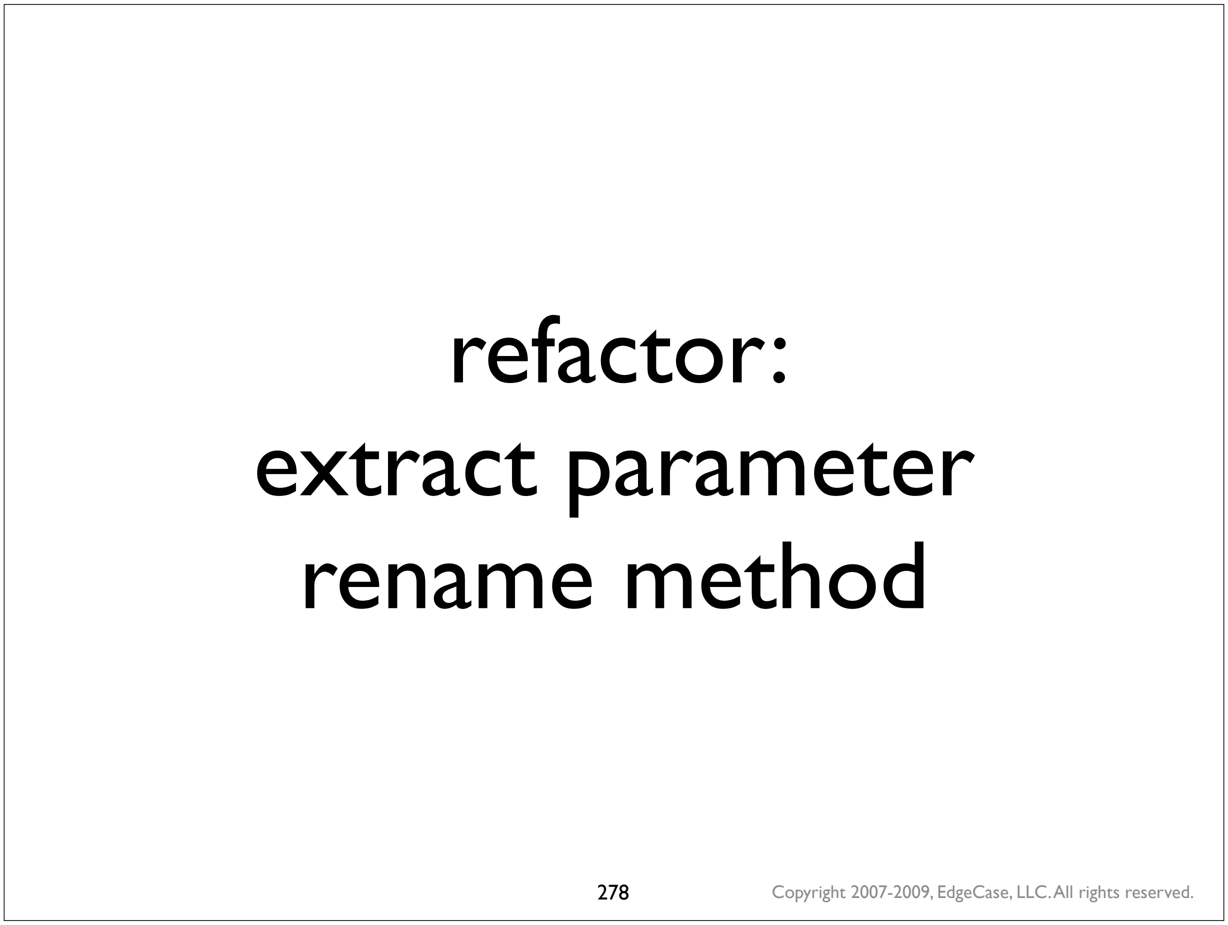
extract method

```ruby
test "should require a score" do
  @player = Player.new(
    :name => "does not matter",
    :score => nil)
  assert_presence_of_score
end

def assert_presence_of_score
  assert ! @player.valid?
  assert @player.errors.on(:score)
  assert_match(/can't be blank/,
    @player.errors.on(:score).to_s)
end
```



```
test "should require a score" do
  @player = Player.new(
    :name => "does not matter",
    :score => nil)
  assert_presence_of_score
end

def assert_presence_of_score
  assert ! @player.valid?
  assert @player.errors.on(:score)
  assert_match(/can't be blank/,
    @player.errors.on(:score).to_s)
end
```

promote to an instance variable



refactor:
extract parameter
rename method

```
def assert_presence_of_score
  assert ! @player.valid?
  assert @player.errors.on(:score)
  assert_match(/can't be blank/,
    @player.errors.on(:score).to_s)
end
```

```
def assert_presence_of(attribute)
  assert ! @player.valid?
  assert @player.errors.on(attribute)
  assert_match(/can't be blank/,
    @player.errors.on(attribute).to_s)
end
```

```
def assert_presence_of(attribute)
  assert ! @player.valid?
  assert @player.errors.on(attribute)
  assert_match(/can't be blank/,
    @player.errors.on(attribute).to_s)
end
```

```
test "should require a name" do
  @player = Player.new(
    :name => nil,
    :score => 10)
  assert_presence_of :name
end

test "should require a score" do
  @player = Player.new(
    :name => "does not matter",
    :score => nil)
  assert_presence_of :score
end
```

refactor:
extract method

```
def valid_options_without(attribute)
  {
   :name => "John Doe",
   :score => 10
  }.merge(attribute => nil)
end
```

```
test "should require a name" do
  @player = Player.new(
    valid_options_without(:name))

  assert_presence_of :name
end

test "should require a score" do
  @player = Player.new(
    valid_options_without(:score))

  assert_presence_of :score
end
```

refactor:
extract method

```ruby
def valid_player_without(attribute)
  @player = Player.new(
    valid_options_without(attribute))
end
```

```ruby
test "should require a name" do
  valid_player_without :name

  assert_presence_of :name
end

test "should require a score" do
  valid_player_without :score

  assert_presence_of :score
end
```

validates_uniqueness_of(:name)

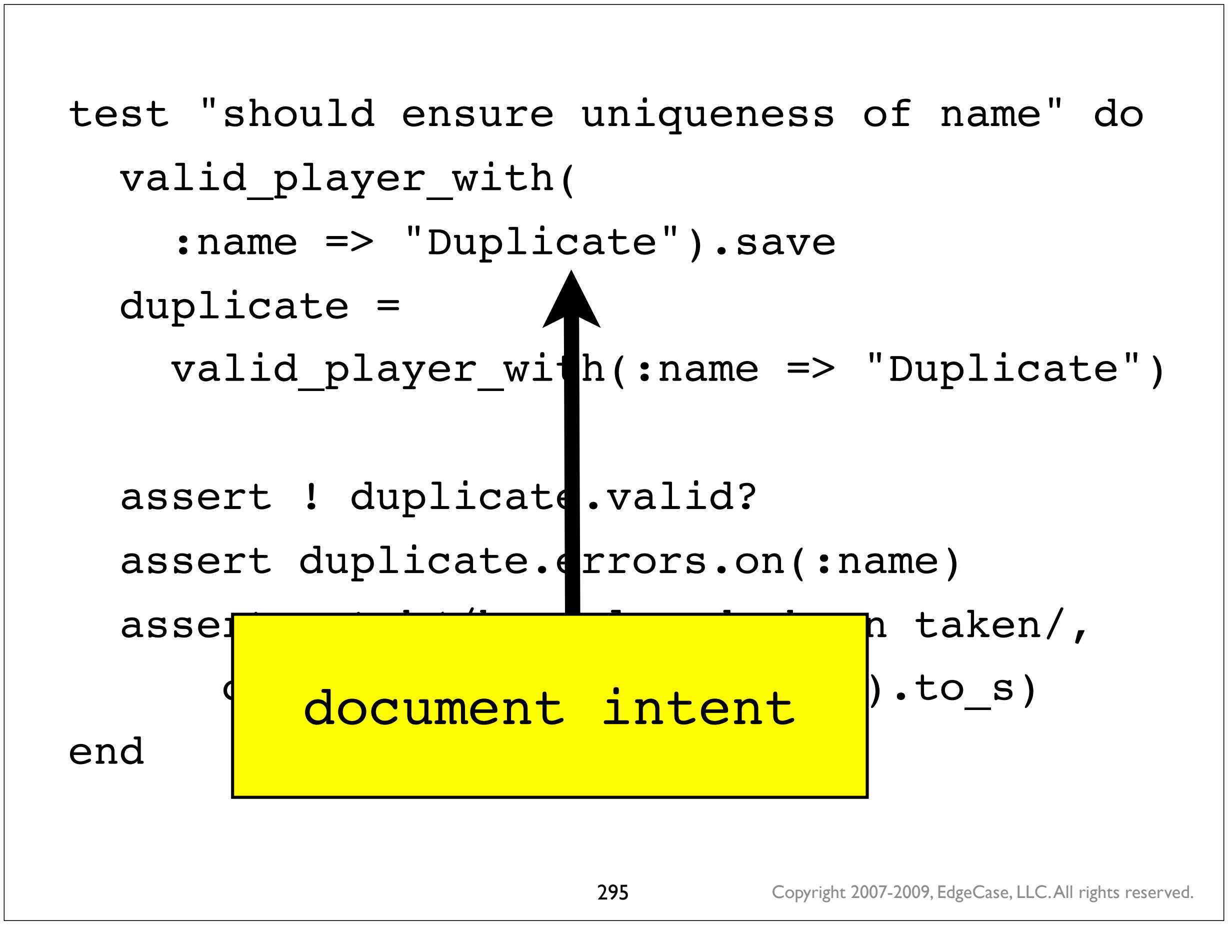
refactor:
extract method

```ruby
def valid_options_without(attribute)
  valid_options_with(attribute => nil)
end

def valid_options_with(attribute)
  {
   :name => "John Doe",
   :score => 10
  }.merge(attribute)
end
```

refactor:
extract method

```
def valid_player_without(attribute)
  @player = Player.new(
    valid_options_without(attribute))
end

def valid_player_with(attribute)
  @player = Player.new(
    valid_options_with(attribute))
end
```

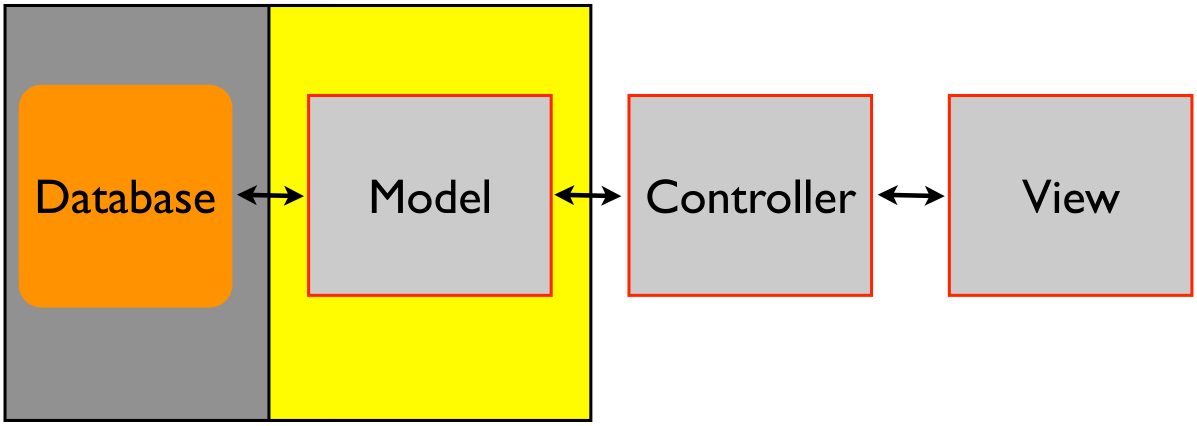
```
test "should ensure uniqueness of name" do
  valid_player_with(
    :name => "Duplicate").save
  duplicate =
    valid_player_with(:name => "Duplicate")

  assert ! duplicate.valid?
  assert duplicate.errors.on(:name)
  assert_match(/has already been taken/,
      duplicate.errors.on(:name).to_s)
end
```



```
test "should ensure uniqueness of name" do
  valid_player_with(
    :name => "Duplicate").save
  duplicate =
    valid_player_with(:name => "Duplicate")

  assert ! duplicate.valid?
  assert duplicate.errors.on(:name)
  asser                          n taken/,
        c                         ).to_s)
end
```

document intent

here is the grey area



```
test "should ensure uniqueness of name" do
  valid_player_with(
    :name => "Duplicate").save
  duplicate =
    valid_player_with(:name => "Duplicate")

  assert ! d
  assert du
  assert_match(/has already been taken/,
      duplicate.errors.on(:name).to_s)
end
```

we need one to be
in the database

refactor:
extract method

```
def assert_presence_of(attribute)
  assert_validation_with_message(
    /can't be blank/, attribute)
end

def assert_validation_with_message(message, attribute)
  assert ! @player.valid?
  assert @player.errors.on(attribute)
  assert_match(message,
    @player.errors.on(attribute).to_s)
end
```

```
test "should ensure uniqueness of name" do
  valid_player_with(
    :name => "Duplicate").save
  duplicate =
    valid_player_with(:name => "Duplicate")

  assert_uniqueness_of(:name)
end

def assert_uniqueness_of(attribute)
  assert_validation_with_message(
    /has already been taken/, attribute)
end
```

remember ...

- Test the **behavior** of your object

  not

- the framework

remember ...

- Refactor your tests **mercilessly**

more on that later ...

Creating Test Data

why not use fixtures?

fixtures are evil!

test_helper.rb

```
# find this and comment it out
# fixtures :all
```

remember this?

```
def valid_options_with(attribute)
  {
   :name => "John Doe",
   :score => 10
  }.merge(attribute)
end
```

Faker

faker.rubyforge.org

Faker

```
$ gem install faker
```

test_helper.rb

```
require 'faker'
```

now this

```
def valid_options_with(attribute)
  {
   :name => Faker::Name.name,
   :score => 10
  }.merge(attribute)
end
```

all kinds of options

```
Faker::Name.name
Faker::Address.street_address
Faker::Internet.email
Faker::PhoneNumber.phone_number
```

Factory Girl

http://github.com/thoughtbot/factory_girl

create

```
test/factories.rb
     or
spec/factories.rb
     or
test/factories/*.rb
     or
spec/factories/*.rb
```

test_helper.rb

```
require 'faker'
require 'factory_girl'
```

test/factories.rb

```
Factory.define :player do |p|
  p.name Faker::Name.name
  p.score 10
end
```

now change this

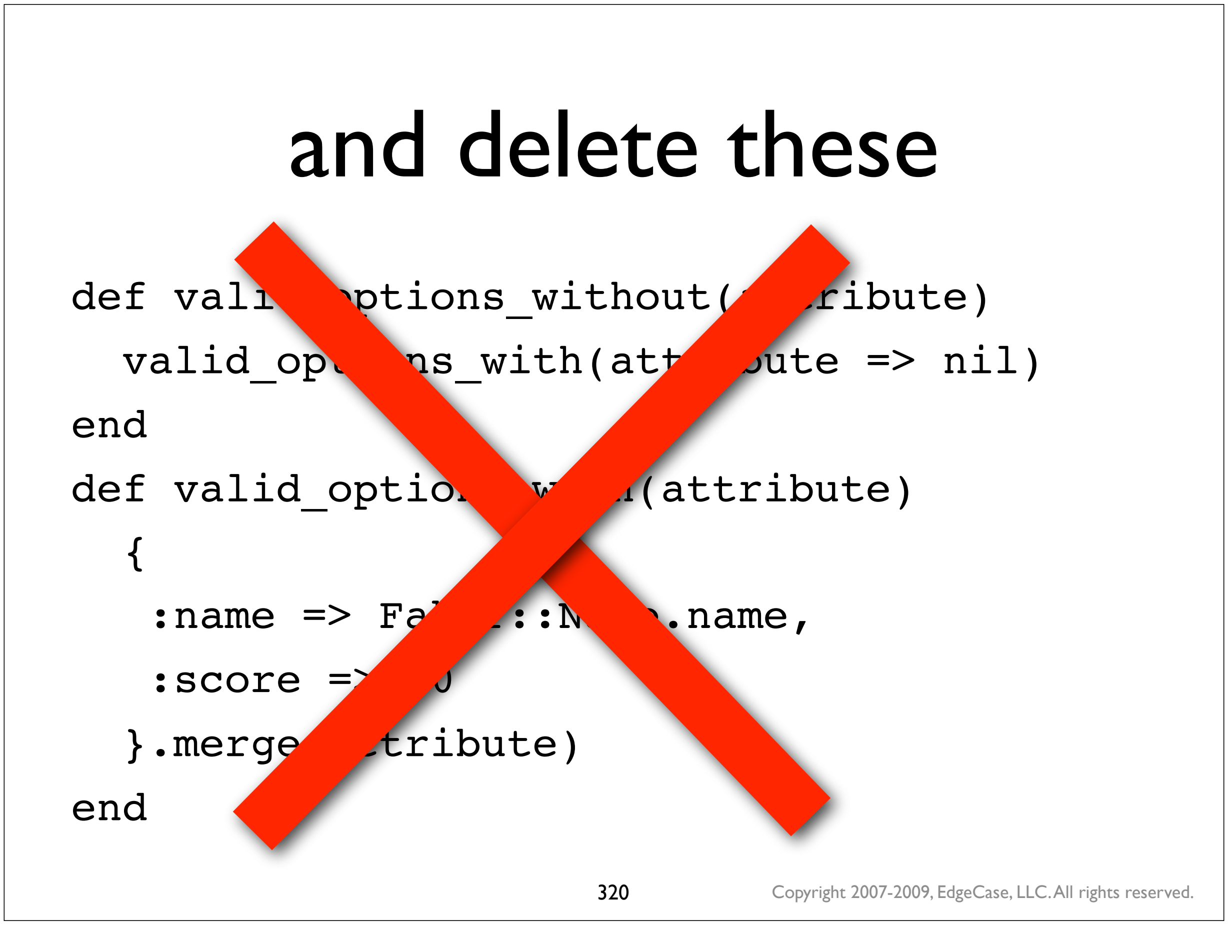
```
def valid_player_without(attribute)
  @player = Player.new(
    valid_options_without(attribute))
end
def valid_player_with(attribute)
  @player = Player.new(
    valid_options_with(attribute))
end
```

to this

```
def valid_player_without(attribute)
  @player = Factory.build(
    :player, attribute => nil)
end
def valid_player_with(attribute)
  @player = Factory.build(
    :player, attribute)
end
```

and delete these

```
def valid_options_without(attribute)
  valid_options_with(attribute => nil)
end
def valid_options_with(attribute)
  {
   :name => Faker::Name.name,
   :score => 10
  }.merge(attribute)
end
```



and delete these

```
def valid_options_without(attribute)
  valid_options_with(attribute => nil)
end
def valid_options_with(attribute)
  {
   :name => Faker::Name.name,
   :score => 10
  }.merge(attribute)
end
```

factory_girl

```
Factory.build :player
Factory.build :player, :name => nil

Factory.create :player

Factory :player   # defaults to create
```

lazy initialization

```
Factory.define :user do |u|
  ...
  u.activation_code { User.generate_code }
end
```

lazy initialization

```
Factory.define :user do |u|
  ...
  u.activation_code { User.generate_code }
end
```

will generate at the time
u.activation_code is called

relationships

```
Factory.define :game do |u|
  ...
  u.players [Factory.build(:player),
             Factory.build(:player),
             Factory.build(:player)]
end
```

Test Driven Development with Rails

Shoulda for Rails

Model Testing

Put this in test helper

```
require “shoulda/rails”
```

Model Macros

Validation Macros

```
def should_validate_presence_of(*attributes)
def should_require_attributes(*attributes)
def should_validate_uniqueness_of(*attributes)
def should_require_unique_attributes(*attributes)
def should_ensure_length_in_range(attribute, range, opts = {})
def should_ensure_length_at_least(attribute, min_length, opts = {})
def should_ensure_length_is(attribute, length, opts = {})
def should_ensure_value_in_range(attribute, range, opts = {})
def should_validate_numericality_of(*attributes)
def should_only_allow_numeric_values_for(*attributes)
def should_validate_acceptance_of(*attributes)
```

Association Macros

```
def should_have_many(*associations)
def should_have_one(*associations)
def should_have_and_belong_to_many(*associations)
def should_belong_to(*associations)
```

Database Macros

```
def should_have_db_columns(*columns)
def should_have_indices(*columns)
def should_have_named_scope(scope_call, find_options = nil)
```

Misc Macros

```
def should_allow_mass_assignment_of(*attributes)
def should_not_allow_mass_assignment_of(*attributes)
def should_protect_attributes(*attributes)
def should_have_readonly_attributes(*attributes)
def should_not_allow_values_for(attribute, *bad_values)
def should_allow_values_for(attribute, *good_values)
def should_have_class_methods(*methods)
def should_have_instance_methods(*methods)
```

Model Macros

```
context “A Player” do
  should_validate_presence_of :name
  should_validate_uniqueness_of :name
  should_validate_presence_of :score

  ... (other tests) ...
end
```

Custom Assertions

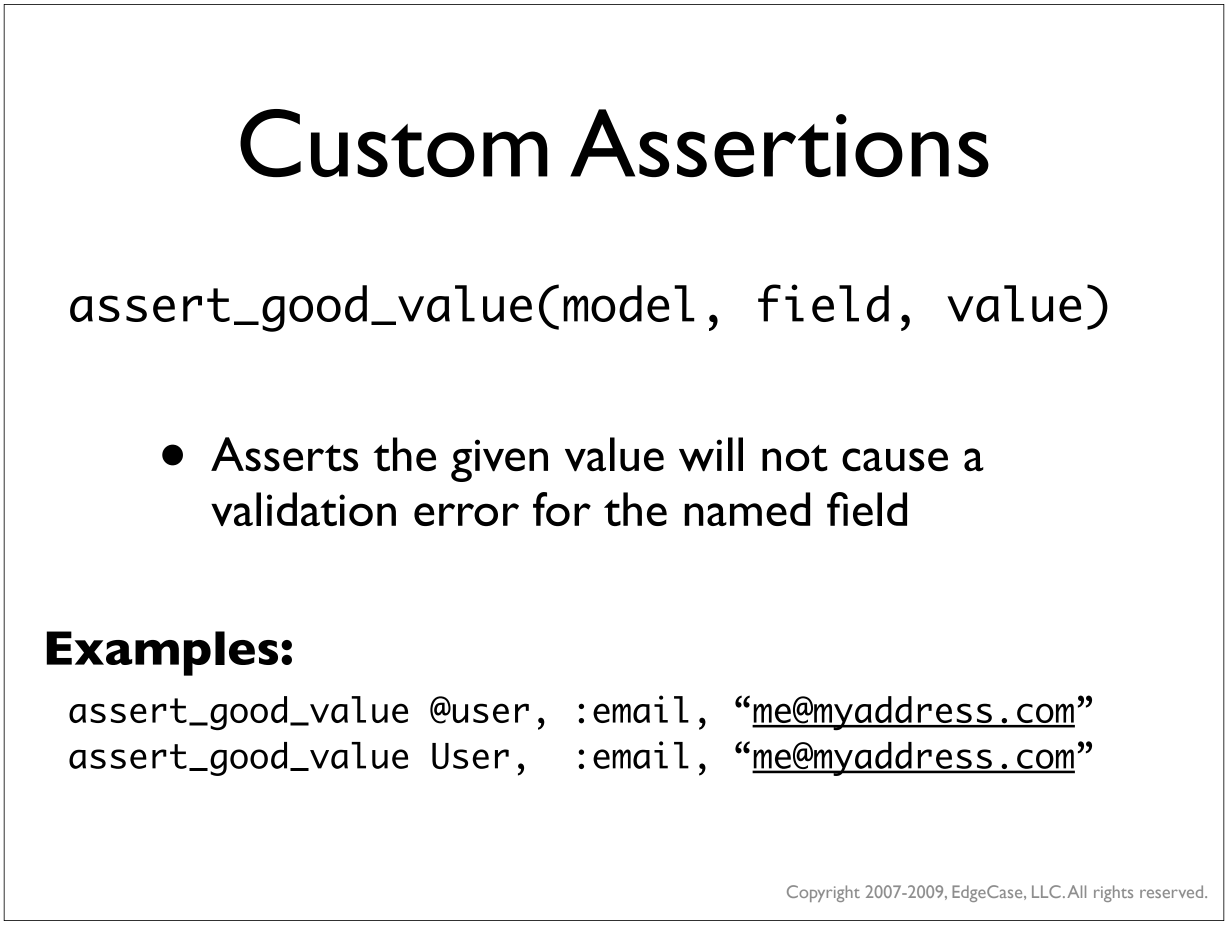
```
assert_save(model_object)
```

- Saves the model
- Asserts the save was successful

Custom Assertions

```
assert_valid(model_object)
```

- Asserts the model object is valid
  - (without saving)



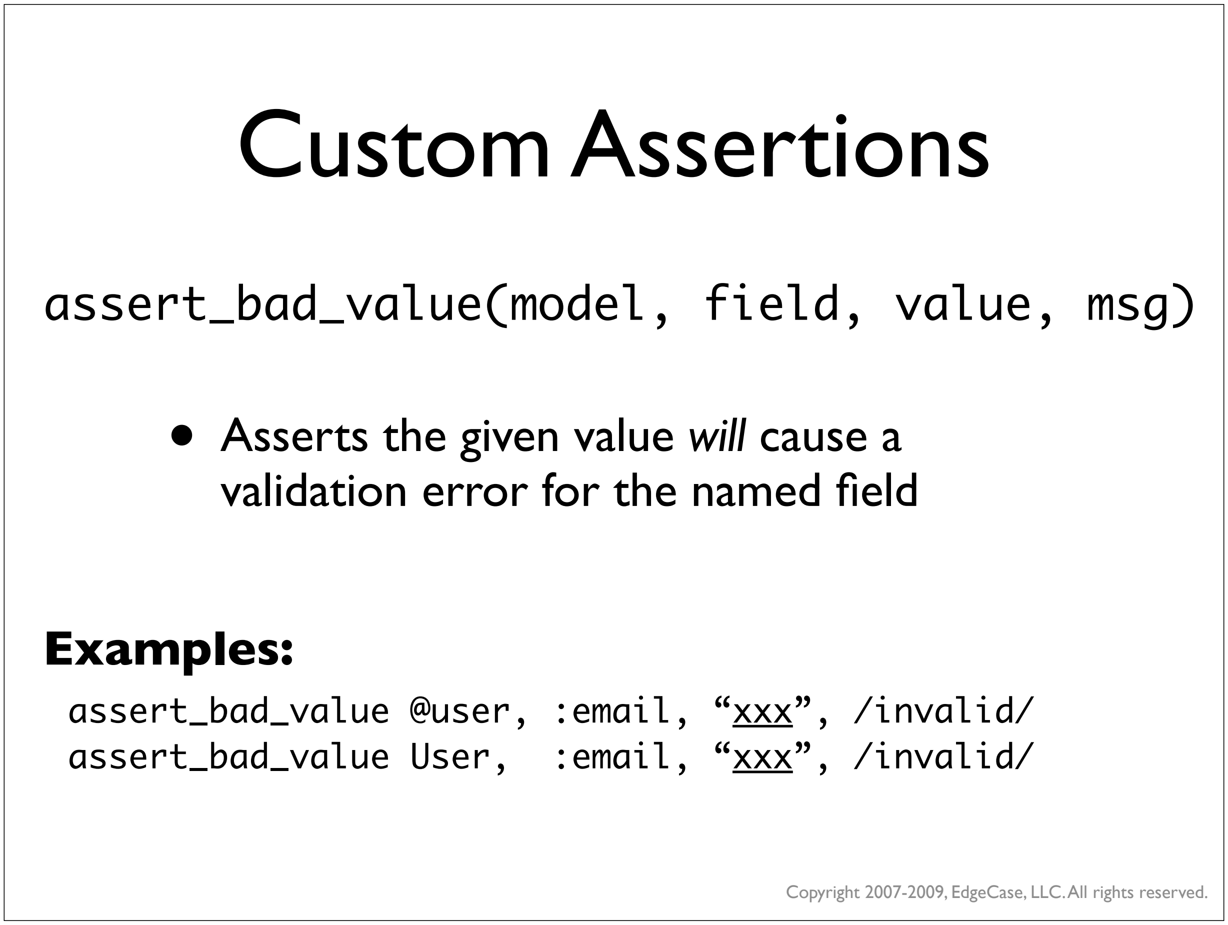
Custom Assertions

```
assert_good_value(model, field, value)
```

- Asserts the given value will not cause a validation error for the named field

Examples:

```
assert_good_value @user, :email, “me@myaddress.com”
assert_good_value User,  :email, “me@myaddress.com”
```



Custom Assertions

```
assert_bad_value(model, field, value, msg)
```

- Asserts the given value *will* cause a validation error for the named field

Examples:

```
assert_bad_value @user, :email, “xxx”, /invalid/
assert_bad_value User,  :email, “xxx”, /invalid/
```

Matchers ??

Matchers

Instead of writing this:

```
context “A RoomType” do
  should_validate_prescence_of :rack_rate
  should_validate_prescence_of :name
  should_validate_uniqueness_of :name
end
```

Matchers

You can write this:

```
require ‘shoulda/active_record/matchers’

context “A RoomType” do
  should validate_prescence_of(:rack_rate)
  should validate_prescence_of(:name)
  should validate_uniqueness_of(:name)
end
```

Matchers

Added Require

You can write this:

```
require ‘shoulda/active_record/matchers’

context “A RoomType” do
  should validate_prescence_of(:rack_rate)
  should validate_prescence_of(:name)
  should validate_uniqueness_of(:name)
end
```

Added Parens

Removed Underscore

Shoulda Docs

http://www.thoughtbot.com/projects/shoulda

Lab 6: Game and Player Models

Lab 6: Game and Player

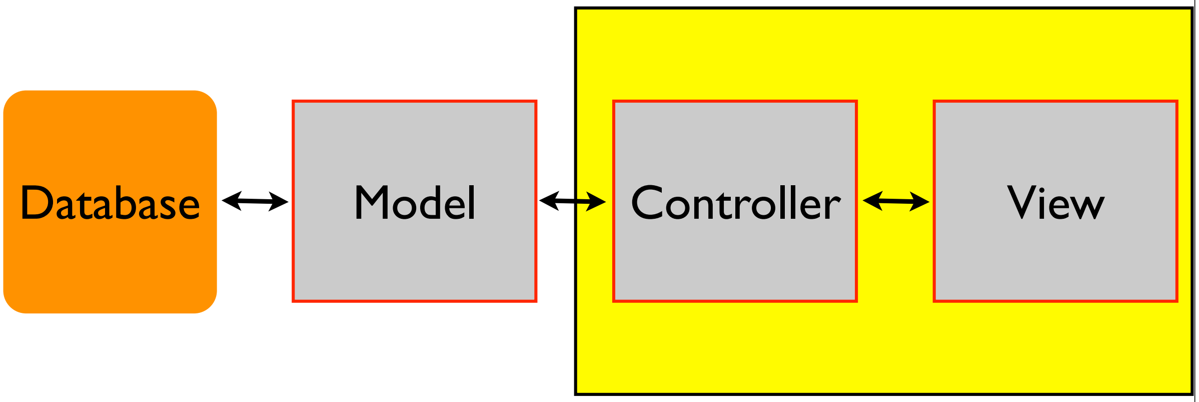
```
Player - name, score, email
Game - no attributes (timestamps)
```

Lab 6: Game and Player

```
Player:
all fields required
score must be a number
email must be valid
```

Testing Controllers

Unit Tests



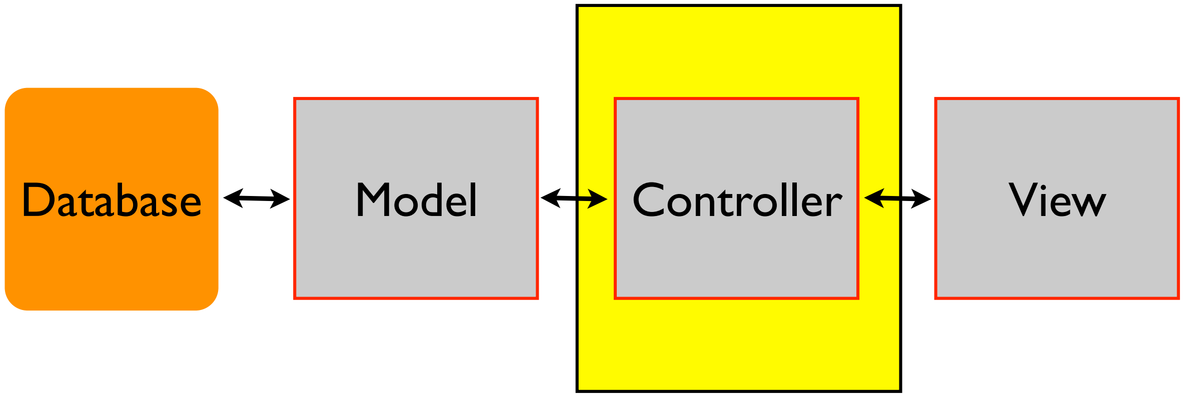
Database

Model

Controller

View

Unit Tests



Database ↔ Model ↔ Controller ↔ View

Other Issues with Controllers

- Controllers are closely tied to the framework
- They have complex interactions with the rest of rails that makes it "interesting" to attempt to decouple controllers for testing.

So, Let’s Create a Controller

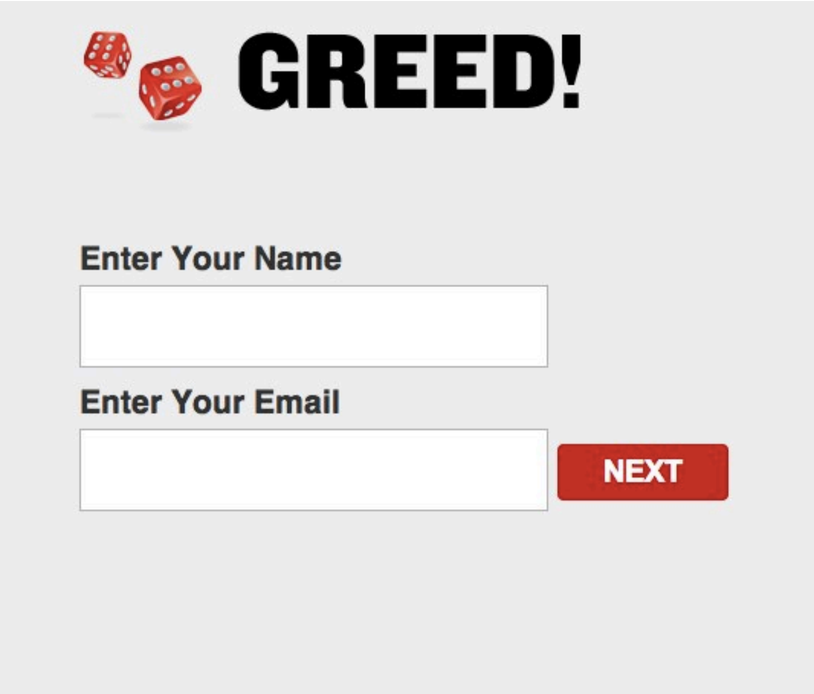
```
$ script/generate controller games
      exists  app/controllers/
      exists  app/helpers/
      create  app/views/games
      exists  test/functional/
      create  test/unit/helpers/
      create  app/controllers/games_controller.rb
      create  test/functional/games_controller_test.rb
      create  app/helpers/games_helper.rb
      create  test/unit/helpers/games_helper_test.rb
```

Initial Controller Test

```
require 'test_helper'

class GamesControllerTest < ActionController::TestCase
  # Replace this with your real tests.
  test "the truth" do
    assert true
  end
end
```

Controller User Story

- The “new” action should create a new game
- It will ask the user for their name and email



Our First Test

```
  test "new sets view variables" do
    get :new

    assert_response :success
    assert_template "new"
  end
```

Our First Test

```
  test "new sets view variables" do
    get :new

    assert_response :success
    assert_template "new"
end
```

- Simulates an HTTP "GET" operation
- Available: get, post, put, delete, head

Our First Test

```
test "new sets view variables" do
  get :new

  assert_response :success
  assert_template "new"
end
```

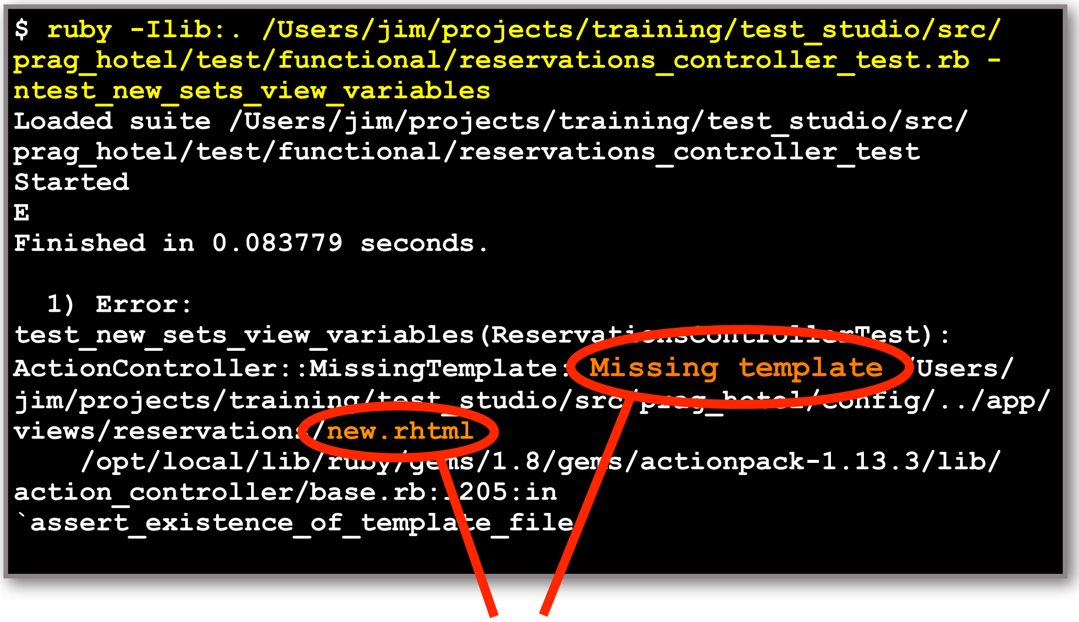
- Checks on the status of the response
- Available: :success (200), :redirect (3xx), :missing (404), :error (5xx)
- Explicit numeric values are allowed

Our First Test

```
test "new sets view variables" do
  get :new

  assert_response :success
  assert_template "new"
  assert assigns(:reservation)
  assert assigns(:availability)
end
```

- Checks the view selected for rendering

Running Our First Test



```
$ ruby -Ilib:. /Users/jim/projects/training/test_studio/src/
prag_hotel/test/functional/reservations_controller_test.rb -
ntest_new_sets_view_variables
Loaded suite /Users/jim/projects/training/test_studio/src/
prag_hotel/test/functional/reservations_controller_test
Started
E
Finished in 0.083779 seconds.

  1) Error:
test_new_sets_view_variables(ReservationsControllerTest):
ActionController::MissingTemplate: Missing template /Users/
jim/projects/training/test_studio/src/prag_hotel/config/../app/
views/reservations/new.rhtml
    /opt/local/lib/ruby/gems/1.8/gems/actionpack-1.13.3/lib/
action_controller/base.rb:1205:in
`assert_existence_of_template_file
```

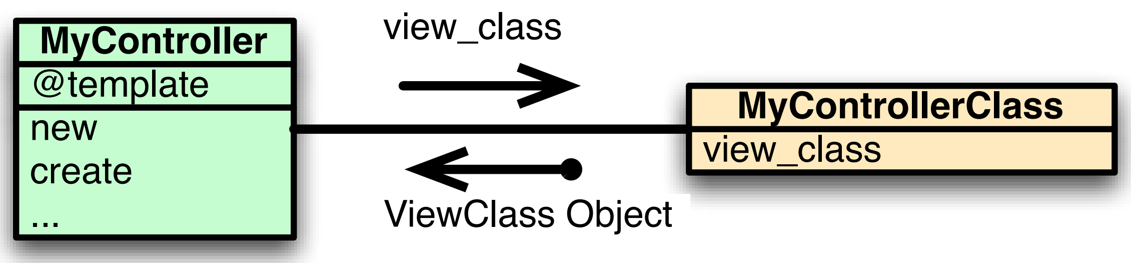
Foiled by a missing view template

we could create `new.html.erb`

or ...
take a step back

Decoupling
View Testing
from
Controller Testing

Quick and Dirty View Stubbing



MyController

@template

new

create

...

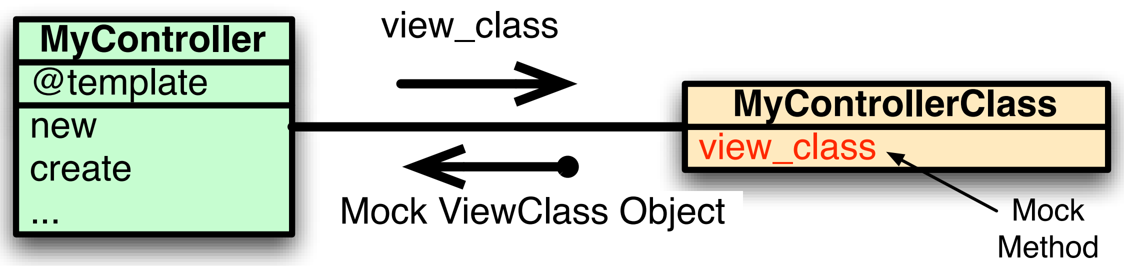
view_class

ViewClass Object

MyControllerClass

view_class

Quick and Dirty View Stubbing



MyController

@template

new

create

...

view_class

Mock ViewClass Object

MyControllerClass

view_class

Mock
Method

Quick and Dirty View Stubbing

```
def stub_view
  view = flexmock("mock view")
  view.should_receive(
    :new => view,
    :assigns => {},
    :file_exists? => true,
    :render_file => true,
    :first_render => true)
  flexmock(@controller.class).
    should_receive(:view_class).
    and_return(view)
end
```

- Put this in your test_helper.rb

Quick and Dirty View Stubbing

```
test "new sets view variables" do
  stub_view

  get :new

  assert_response :success
end
```

- Put this in your test_helper.rb

Quick and Dirty View Stubbing

```
require ‘flexmock/rails’

 test "new sets view variables" do
   should_render_view(“new”)

   get :new

   assert_response :success

  end
```

- Now a standard part of flexmock

Implement “new”

app/controllers/games_controller.rb

```
  def new

  end
```

nothing much to do here

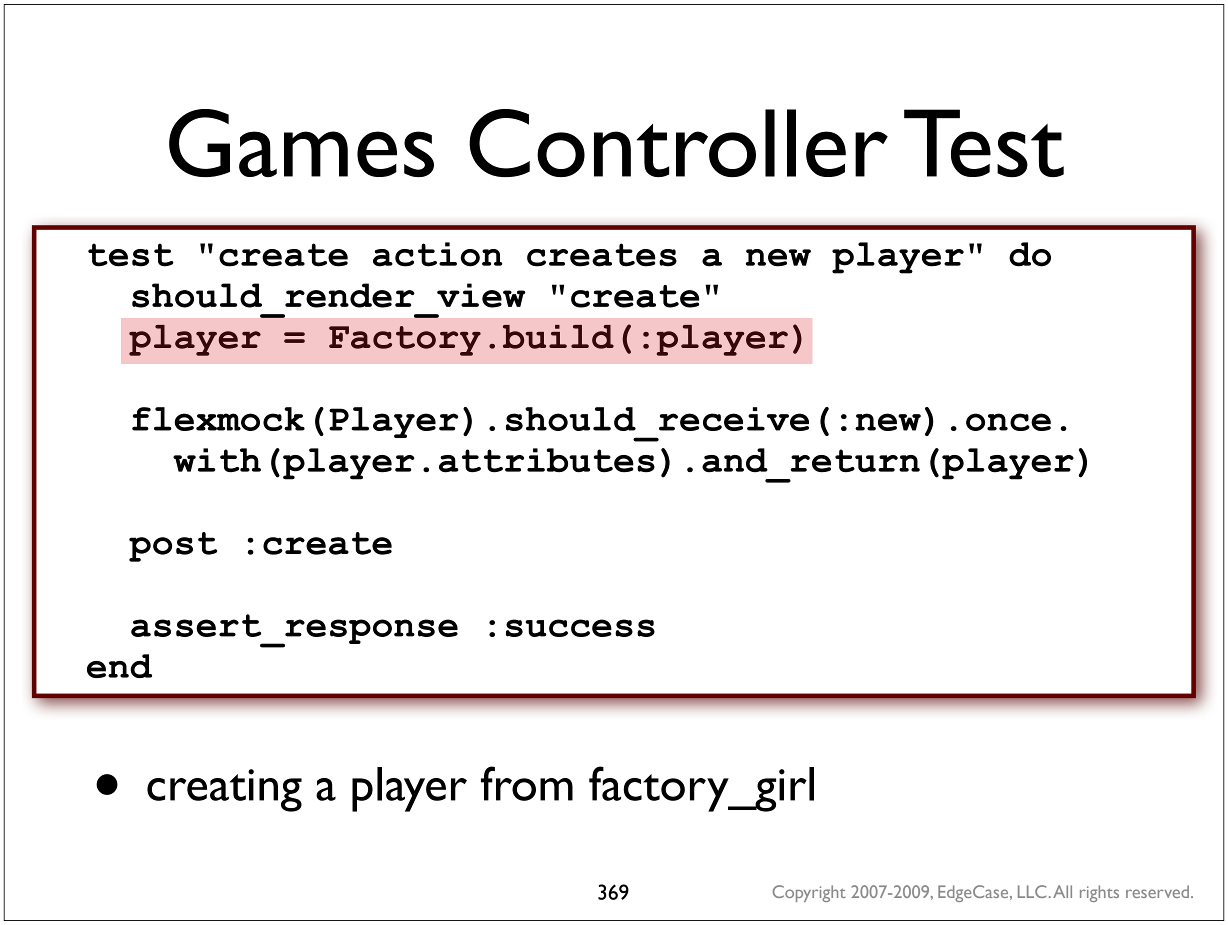
Controller User Story

- A player will be created by the input
- Create will create a game
- A player will be added to the game
- Game id will be saved in the session
- Create will redirect to choose_players

Games Controller Test

```
test "create action creates a new player" do
  should_render_view "create"
  player = Factory.build(:player)

  flexmock(Player).should_receive(:new).once.
    with(player.attributes).and_return(player)

  post :create

  assert_response :success
end
```

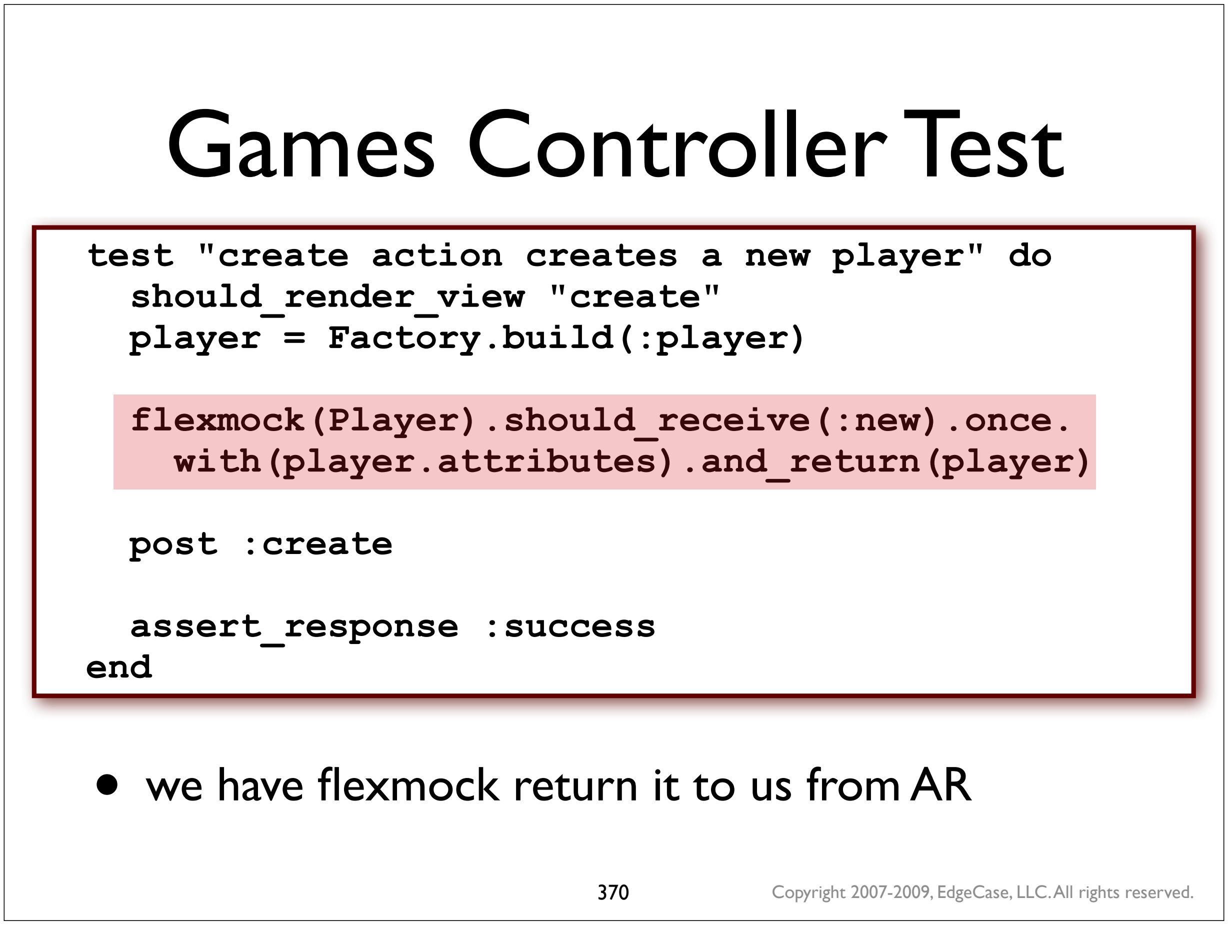
- Just a placeholder



Games Controller Test

```
test "create action creates a new player" do
  should_render_view "create"
  player = Factory.build(:player)

  flexmock(Player).should_receive(:new).once.
    with(player.attributes).and_return(player)

  post :create

  assert_response :success
end
```

- creating a player from factory_girl



Games Controller Test

```
test "create action creates a new player" do
  should_render_view "create"
  player = Factory.build(:player)

  flexmock(Player).should_receive(:new).once.
    with(player.attributes).and_return(player)

  post :create

  assert_response :success
end
```

- we have flexmock return it to us from AR

Games Controller Test

```
test "create action creates a new player" do
  should_render_view "create"
  player = Factory.build(:player)

  flexmock(Player).should_receive(:new).once.
    with(player.attributes).and_return(player)

  post :create

  assert_response :success
end
```

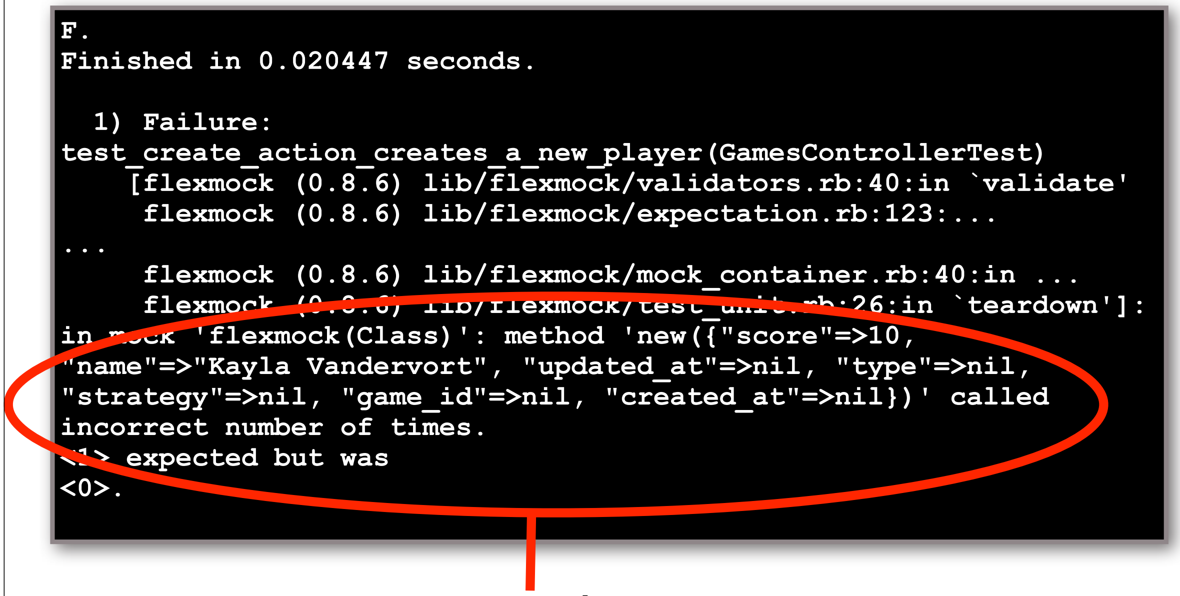
- we make sure it is called

Games Controller Test

```
test "create action creates a new player" do
  should_render_view "create"
  player = Factory.build(:player)

  flexmock(Player).should_receive(:new).once.
    with(player.attributes).and_return(player)

  post :create

  assert_response :success
end
```

- we are not really that concerned with assertion

Run the Test



```
F.
Finished in 0.020447 seconds.

  1) Failure:
test_create_action_creates_a_new_player(GamesControllerTest)
    [flexmock (0.8.6) lib/flexmock/validators.rb:40:in `validate'
     flexmock (0.8.6) lib/flexmock/expectation.rb:123:...
...
     flexmock (0.8.6) lib/flexmock/mock_container.rb:40:in ...
     flexmock (0.8.6) lib/flexmock/test_unit.rb:26:in `teardown']:
in mock 'flexmock(Class)': method 'new({"score"=>10,
"name"=>"Kayla Vandervort", "updated_at"=>nil, "type"=>nil,
"strategy"=>nil, "game_id"=>nil, "created_at"=>nil})' called
incorrect number of times.
<1> expected but was
<0>.
```

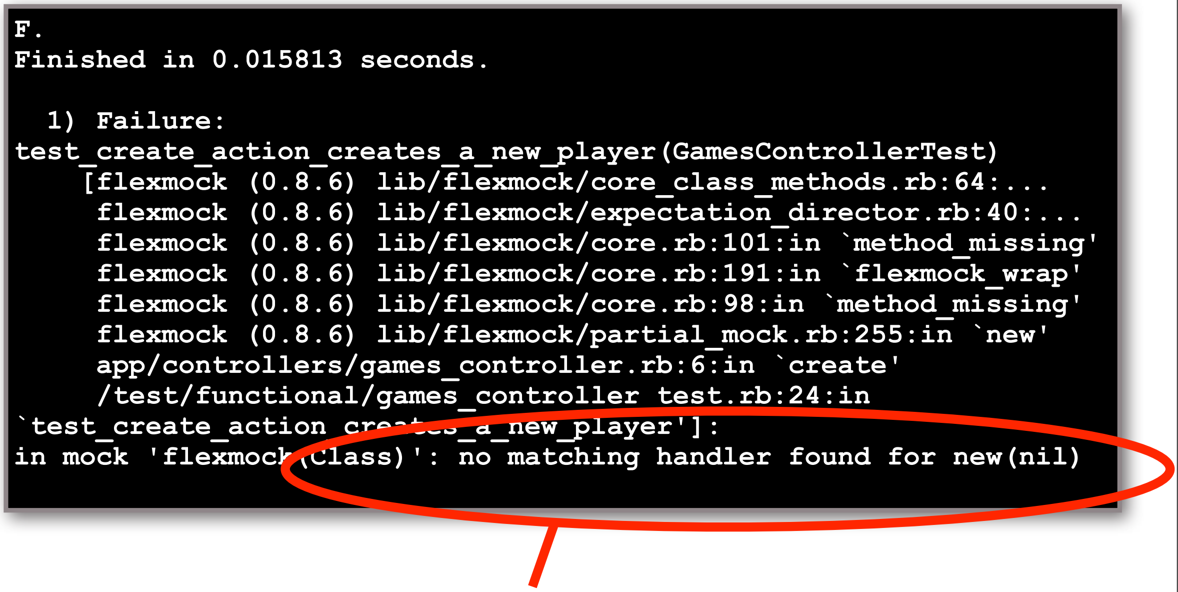
because we haven't implemented it yet

Implement “new”

app/controllers/games_controller.rb

```
def create
 @player = Player.new(params[:player])
end
```

Run the Test



```
F.
Finished in 0.015813 seconds.

  1) Failure:
test_create_action_creates_a_new_player(GamesControllerTest)
    [flexmock (0.8.6) lib/flexmock/core_class_methods.rb:64:...
     flexmock (0.8.6) lib/flexmock/expectation_director.rb:40:...
     flexmock (0.8.6) lib/flexmock/core.rb:101:in `method_missing'
     flexmock (0.8.6) lib/flexmock/core.rb:191:in `flexmock_wrap'
     flexmock (0.8.6) lib/flexmock/core.rb:98:in `method_missing'
     flexmock (0.8.6) lib/flexmock/partial_mock.rb:255:in `new'
     app/controllers/games_controller.rb:6:in `create'
     /test/functional/games_controller_test.rb:24:in
`test_create_action_creates_a_new_player']:
in mock 'flexmock(Class)': no matching handler found for new(nil)
```

its getting a call to new, but not with any parameters

Games Controller Test

```
test "create action creates a new player" do
  should_render_view "create"
  player = Factory.build(:player)

  flexmock(Player).should_receive(:new).once.
    with(player.attributes).and_return(player)

  post :create, :player => player.attributes

  assert_response :success
end
```

- we forgot to pass in the parameters

the assigns hash

Games Controller Test

```
test "create action creates a new player" do
  should_render_view "create"
  player = Factory.build(:player)

  flexmock(Player).should_receive(:new).once.
    with(player.attributes).and_return(player)

  post :create, :player => player.attributes

  assert_response :success
  assert_equal player, assigns(:player)
end
```

- looks up any instance variables set in controller can also check:

```
flash[]
cookies[]
session[]
```

Odds and Ends

Testing Session Data

```
get “login”,
  :user_id => user.id

assert_equal user.id.to_s,
  session[“user_id”]
```

Use `session` to access session data set in the controller.

Testing Session Data

```
get “admin_stuff”,
  { :id => 1 },                   # Params Hash
  { :user_id => @user.id }        # Session Hash
```

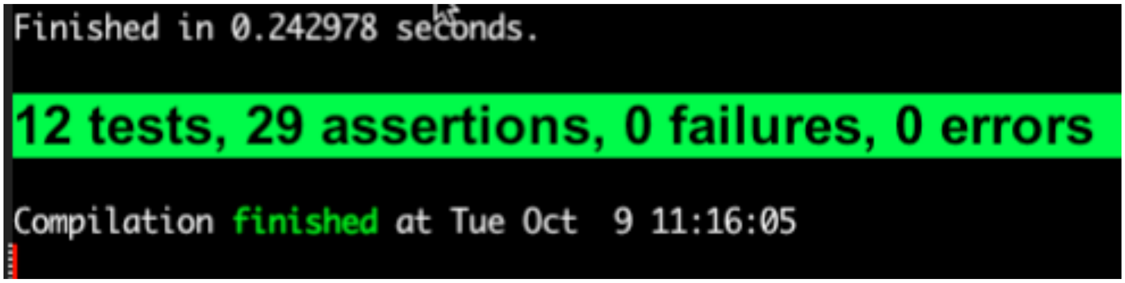
Pass session data to the controller
in a separate hash to `get/put/post/delete`

Testing Session Data

```
def login_as_user
  @user = flexmock(:model, User)
  @request.session[:user_id] = @user.id
end
```

Use `@request.session`
to populate the session before the test.

Done



```
Finished in 0.242978 seconds.

12 tests, 29 assertions, 0 failures, 0 errors

Compilation finished at Tue Oct  9 11:16:05
```

Test Driven Development with Rails

Shoulda for Rails

Controller Testing

Controller Macros

Render Macros

```
def should_render_template(template)
def should_render_with_layout(expected_layout)
def should_render_without_layout
def should_redirect_to(url)
def should_render_a_form
def should_render_page_with_metadata(options)
```

Response Macros

```
def should_respond_with(response)
def should_respond_with_content_type(content_type)
```

Misc Macros

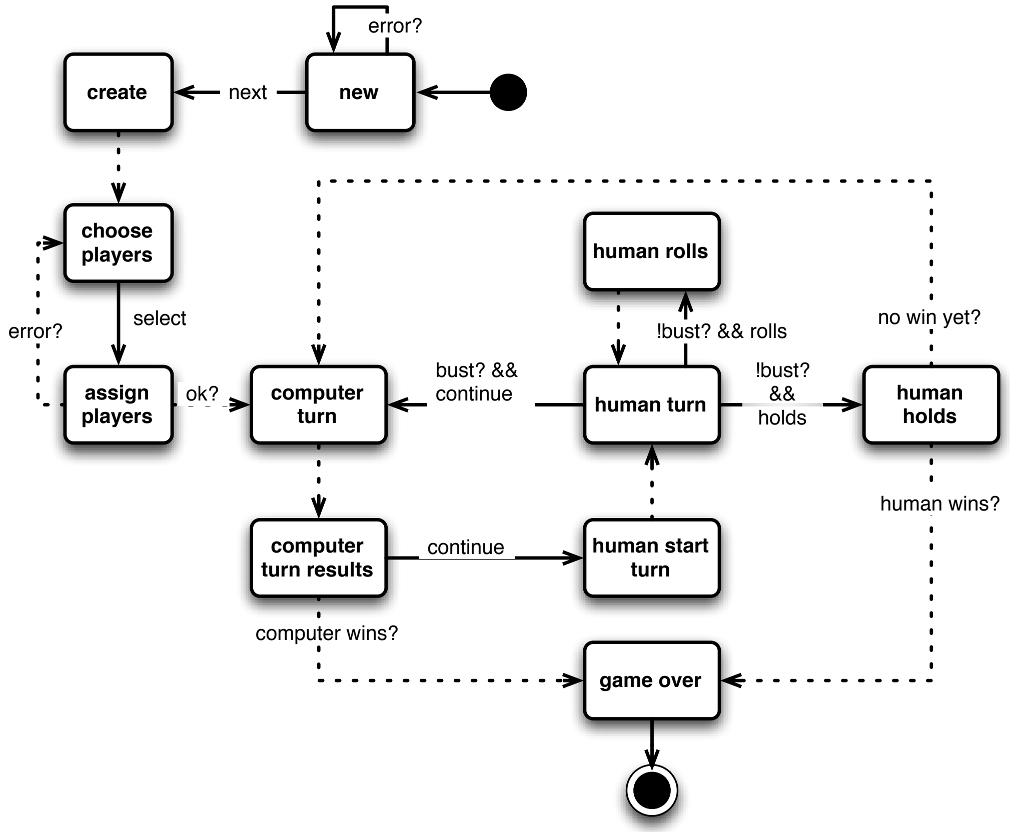
```
def should_set_the_flash_to(val)
def should_not_set_the_flash
def should_filter_params(*keys)
def should_assign_to(*names)
def should_not_assign_to(*names)
def should_route(method, path, options)
def should_return_from_session(key, expected)
```

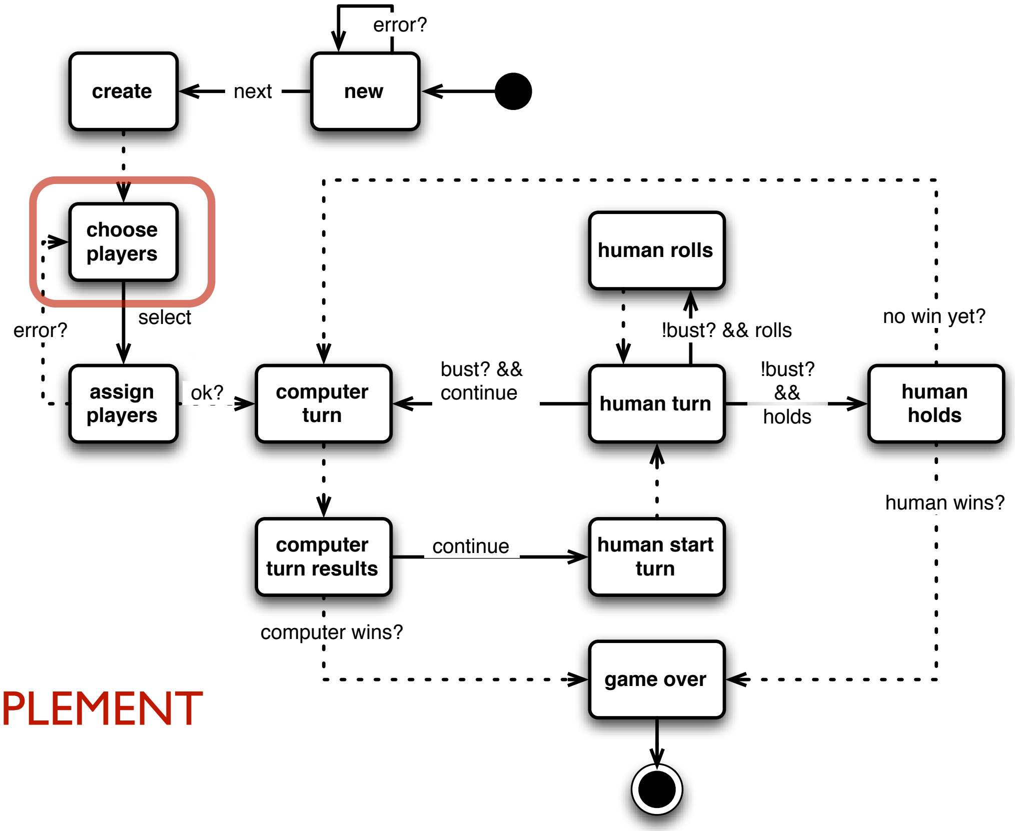
Deprecated Macros

```
def should_be_restful(&blk)
```

(don’t use this one)

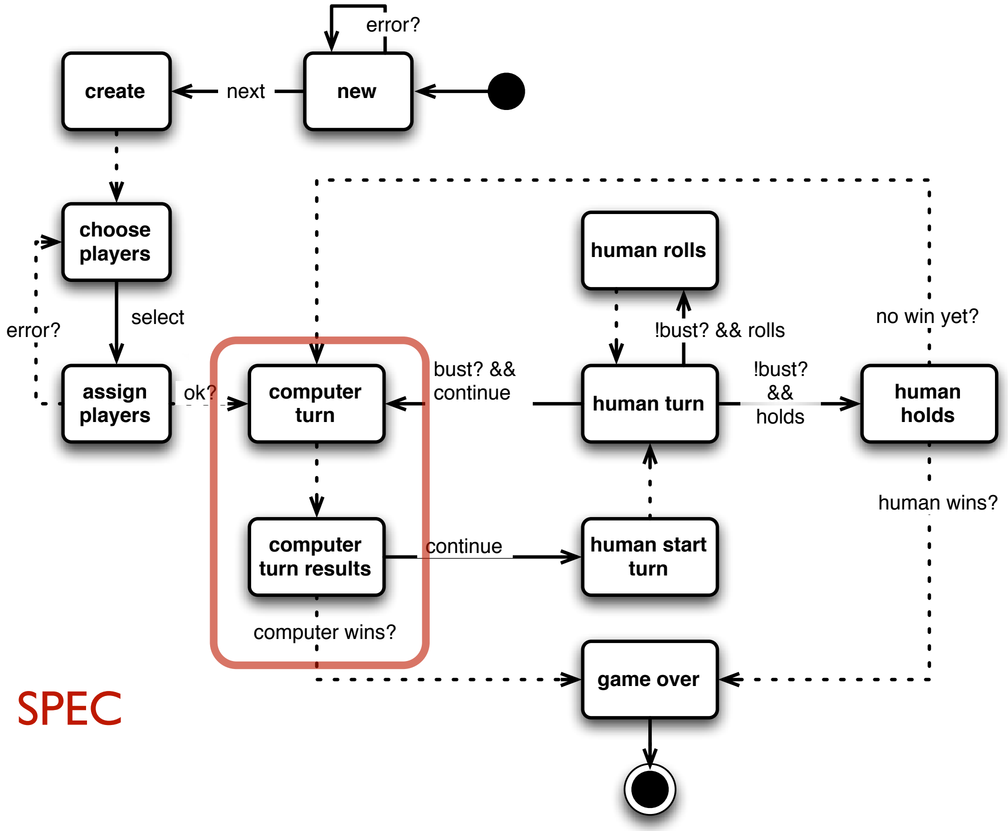
Lab 7: Controller Testing





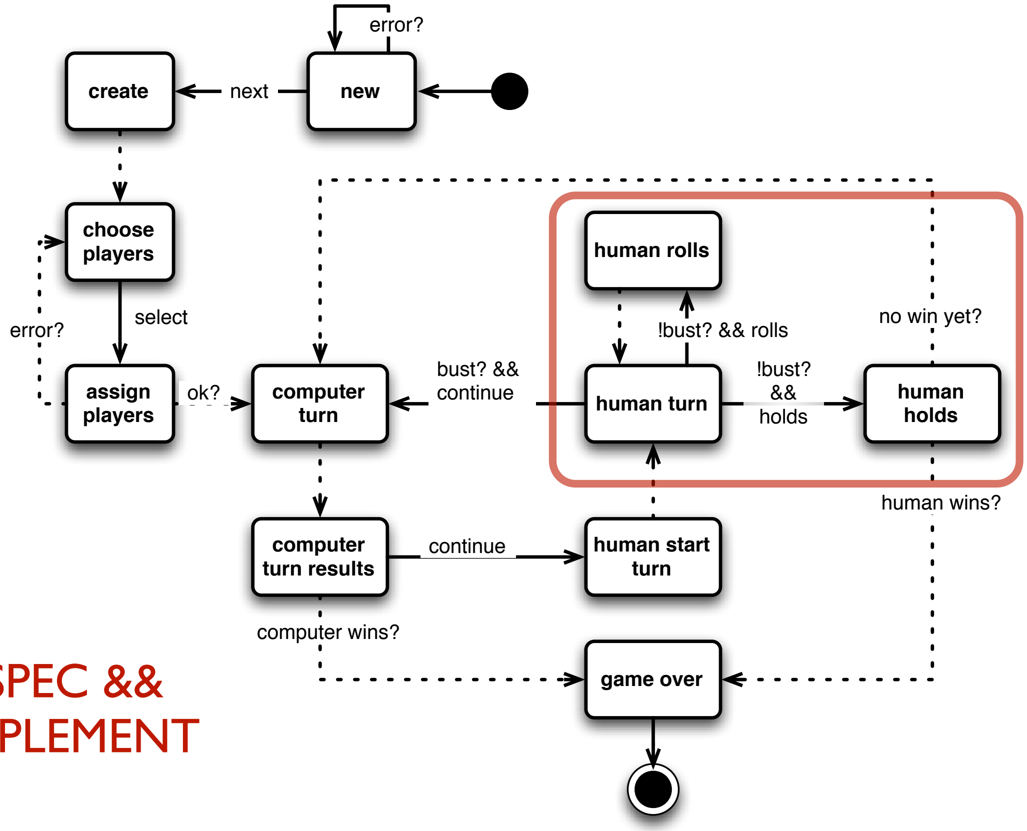
error?
create
next
new
choose players
select
error?
assign players
ok?
computer turn
bust? && continue
human rolls
!bust? && rolls
human turn
!bust? && holds
no win yet?
human holds
human wins?
computer turn results
continue
human start turn
computer wins?
game over

IMPLEMENT



error?
create
next
new
choose players
select
error?
assign players
ok?
computer turn
bust? && continue
human rolls
!bust? && rolls
human turn
!bust? && holds
no win yet?
human holds
human wins?
computer turn results
continue
human start turn
computer wins?
game over

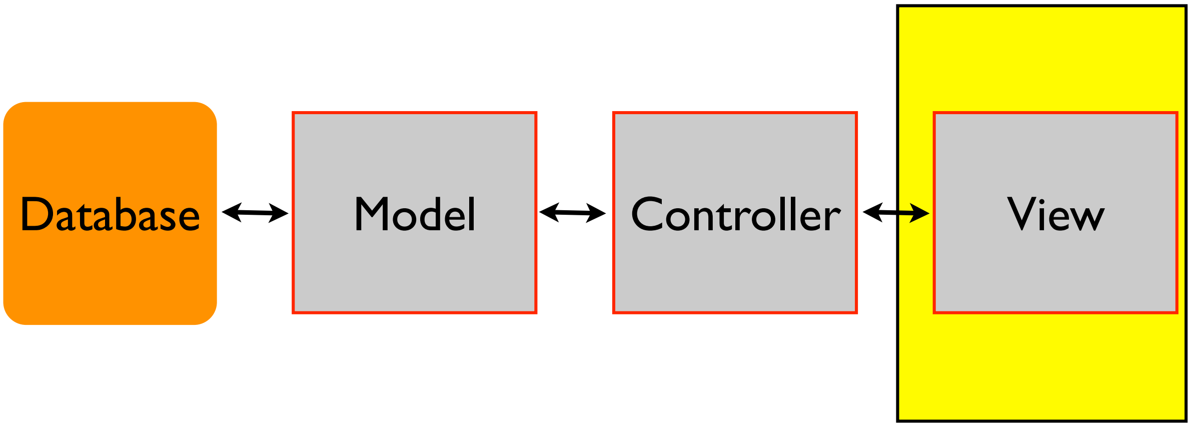
SPEC



SPEC && IMPLEMENT

View Testing

View Tests ?



Database

Model

Controller

View

why?

test behavior in view

test things that can break

not testing framework

```
<%= @reservation.check_in.to_s(:long) -%>
```

testing behavior

```
<% if flash[:warn] %>
  <div class='warning'>
    <%= flash[:warn] %>
  </div>
<% end %>
```

testing things that could break

```
<% form_for :reservation, @reservation...
  Check In Date: <%= date_select("rese...
  <br />
  Check Out Date: <%= date_select("res...
  <br />
  <%= submit_tag 'Check Rate' -%>
<% end -%>
```

The Rails Way

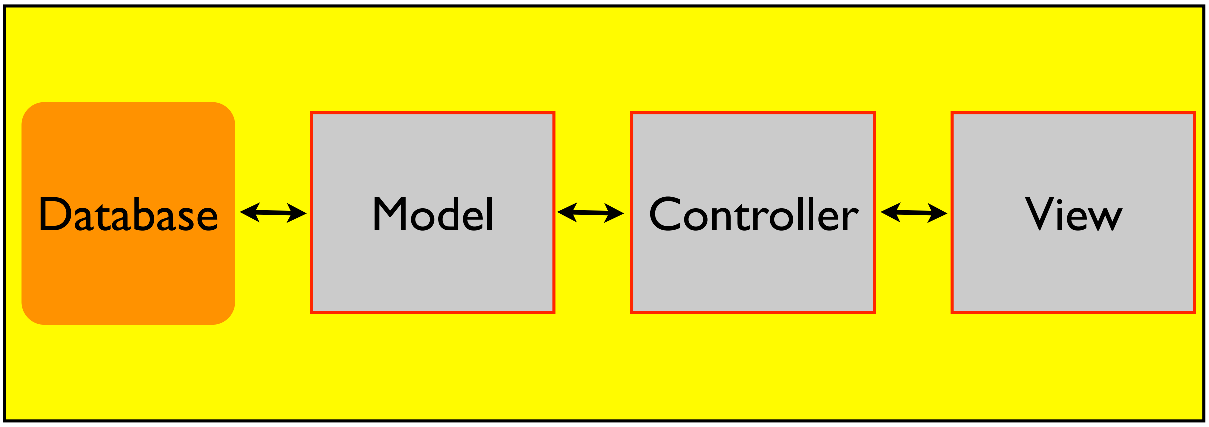
assert_tag

```
def  test_view_should_display_error_...
  check_in = Date.today
  check_out = check_in - 2

  get :new, :reservation =>
     reservation(check_in, check_out)

  assert_tag
    :div,
    :attributes => { :class => "warning" }
end
```

```
def  test_view_should_display_error_...
  check_in = Date.today
  check_out = check_in - 2

  get :new, :reservation =>
     reservation(check_in, check_out)

  assert_tag
   :div,
   :attributes => { :class => "warning" }
end
```

problems

a lot of moving parts



Database ↔ Model ↔ Controller ↔ View

error messages

```
 1) Failure:
test_view_should_display_erro
    [/opt/local/lib/ruby/gems/1.8/gems/action
     /opt/local/lib/ruby/gems/1.8/gems/action
     test/functional/
reservations_controller_expected tag, but no tag
found matching {:tag=>"div", :attributes=>
{:class=>"warnings"}} in:
"<html>\n<head>\n  <script src=\"/javascripts/
prototype.js?1191548071\" type=\"text/javascript
\"></script>\n<script src=\"/javascripts/
effects.js?1191548071\" type=\"text/javascript
\"></script>\n<script src=\"/javascripts/
dragdrop.js?1191548071\" type=\"text/javascript
\"></script>\n<script src=\"/javascripts/
controls.js?1191548071\" type=\"text/javascript
\"></script>\n<script src=\"/javascripts/
application.js?1191548071\" type=\"text/
javascript\`"></script>  <link href=\"/
```

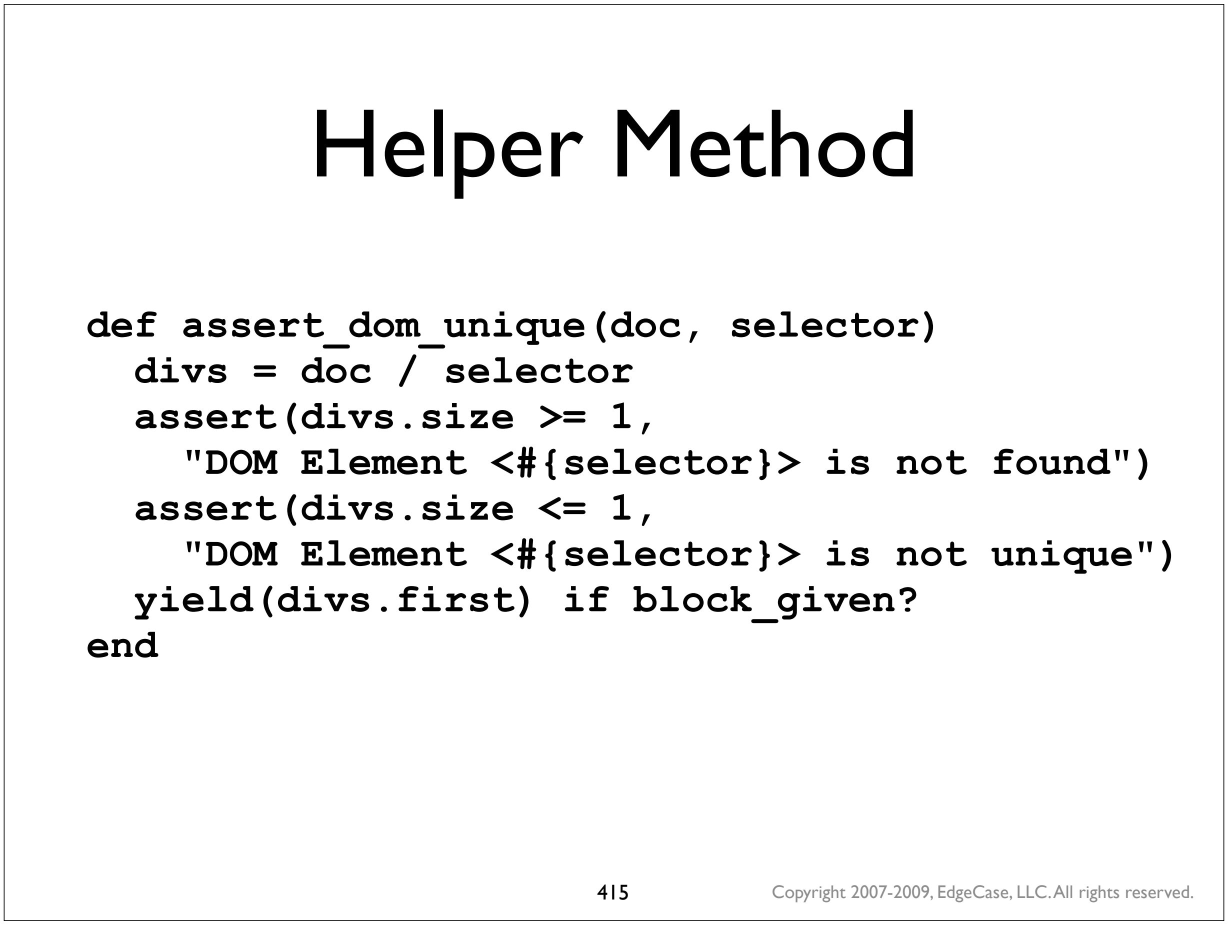
Parsed with REXML

- REXML == Speed::SLOW
- HTML == XHTML

Hpricot

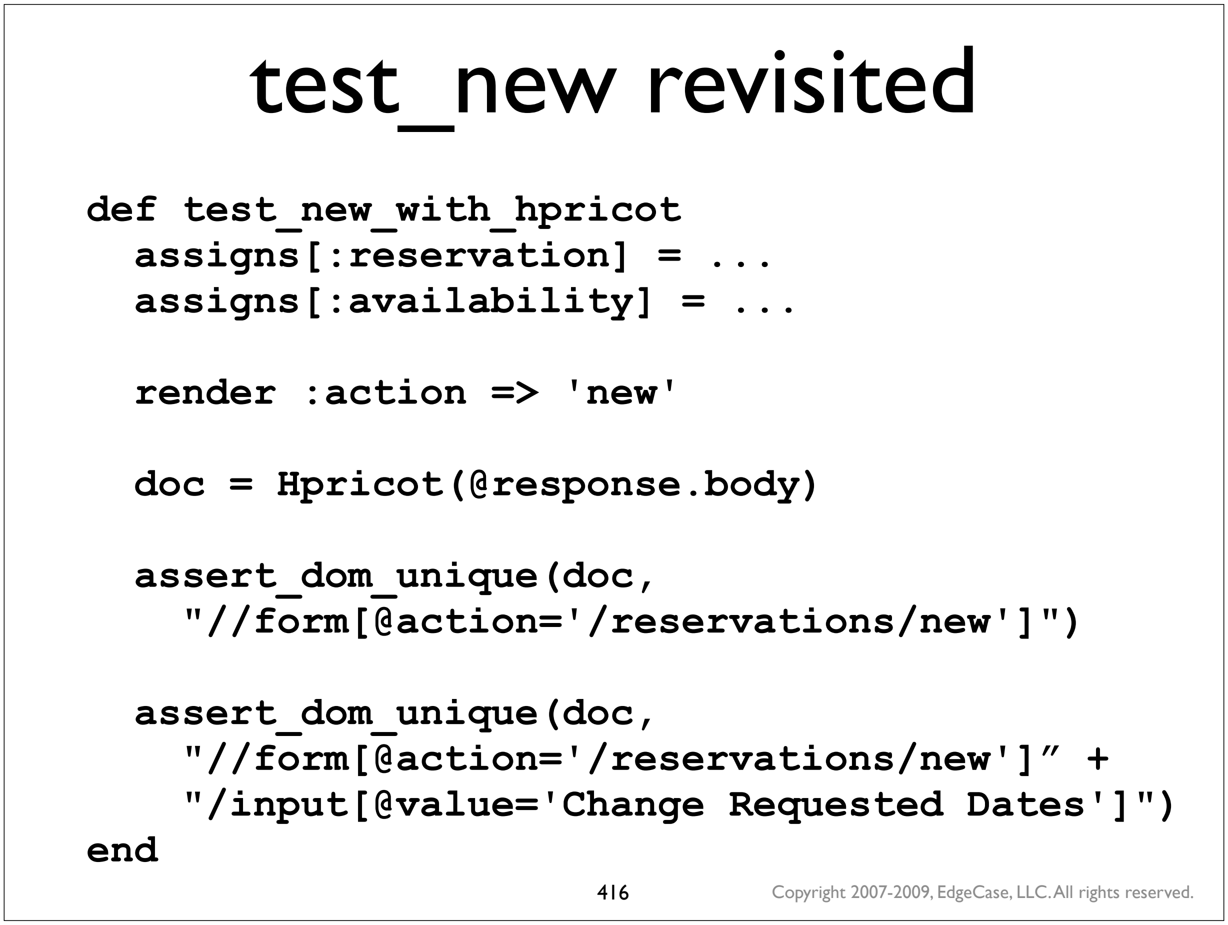
Using Hpricot

```
require ‘hpricot’

html = ‘<div class=”x”><span id=”y”></span></div>’
doc = Hpricot(html)

doc / ‘div’              # list of divs in doc
doc / ‘span’             # list of spans in doc
doc / ‘div’ / ‘span’     # list of nested spans in divs
doc / ‘div/span’         # (same as above)
doc / ‘span[@id=”y”]’    # list of spans with id=y
doc / ‘div[@class=”x”]’  # list of divs with class=x
```



Helper Method

```
def assert_dom_unique(doc, selector)
  divs = doc / selector
  assert(divs.size >= 1,
    "DOM Element <#{selector}> is not found")
  assert(divs.size <= 1,
    "DOM Element <#{selector}> is not unique")
  yield(divs.first) if block_given?
end
```



test_new revisited

```
def test_new_with_hpricot
  assigns[:reservation] = ...
  assigns[:availability] = ...

  render :action => 'new'

  doc = Hpricot(@response.body)

  assert_dom_unique(doc,
    "//form[@action='/reservations/new']")

  assert_dom_unique(doc,
    "//form[@action='/reservations/new']″ +
    "/input[@value='Change Requested Dates']")
end
```

RSpec on Rails

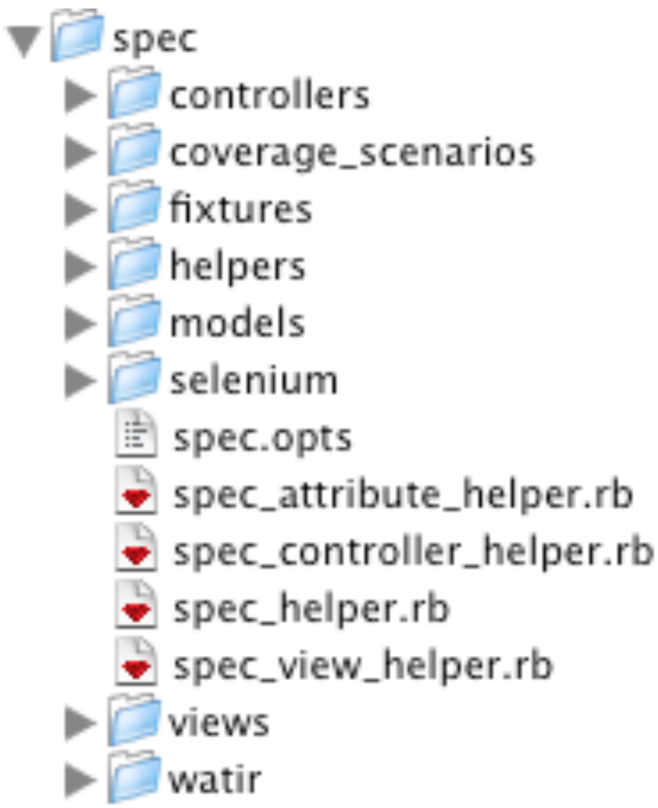
gem

```
$ gem install rspec-rails
```

Test::Rails influence

All-in-one package

- Unit testing (heavy Test::Rails influence)
- Mocking / Stubbing
  - but we don’t like it :-)
- Integration Testing (Spec::UI)



```
▼ spec
    ▶ controllers
    ▶ coverage_scenarios
    ▶ fixtures
    ▶ helpers
    ▶ models
    ▶ selenium
      spec.opts
      spec_attribute_helper.rb
      spec_controller_helper.rb
      spec_helper.rb
      spec_view_helper.rb
    ▶ views
    ▶ watir
```

rake tasks

```
rake app:spec                         # Run all ...
rake spec                             # Run all ...
rake spec:clobber_rcov                # Remove r ...
rake spec:controllers                 # Run the ...
rake spec:db:fixtures:load            # Load fix ...
rake spec:doc                         # Print Sp ...
rake spec:generate_specs_from_yaml    # Converts ...
rake spec:helpers                     # Run the  ...
rake spec:lib                         # Run the  ...
rake spec:models                      # Run the  ...
rake spec:plugin_doc                  # Print Sp ...
rake spec:plugins                     # Run the  ...
rake spec:plugins:rspec_on_rails      # Runs the ...
rake spec:rcov                        # Run all  ...
rake spec:server:restart              # reload   ...
rake spec:server:start                # start    ...
rake spec:server:stop                 # stop     ...
rake spec:translate                   # Translat ...
```

rcov integration

time to play!

RCov Coverage Tool

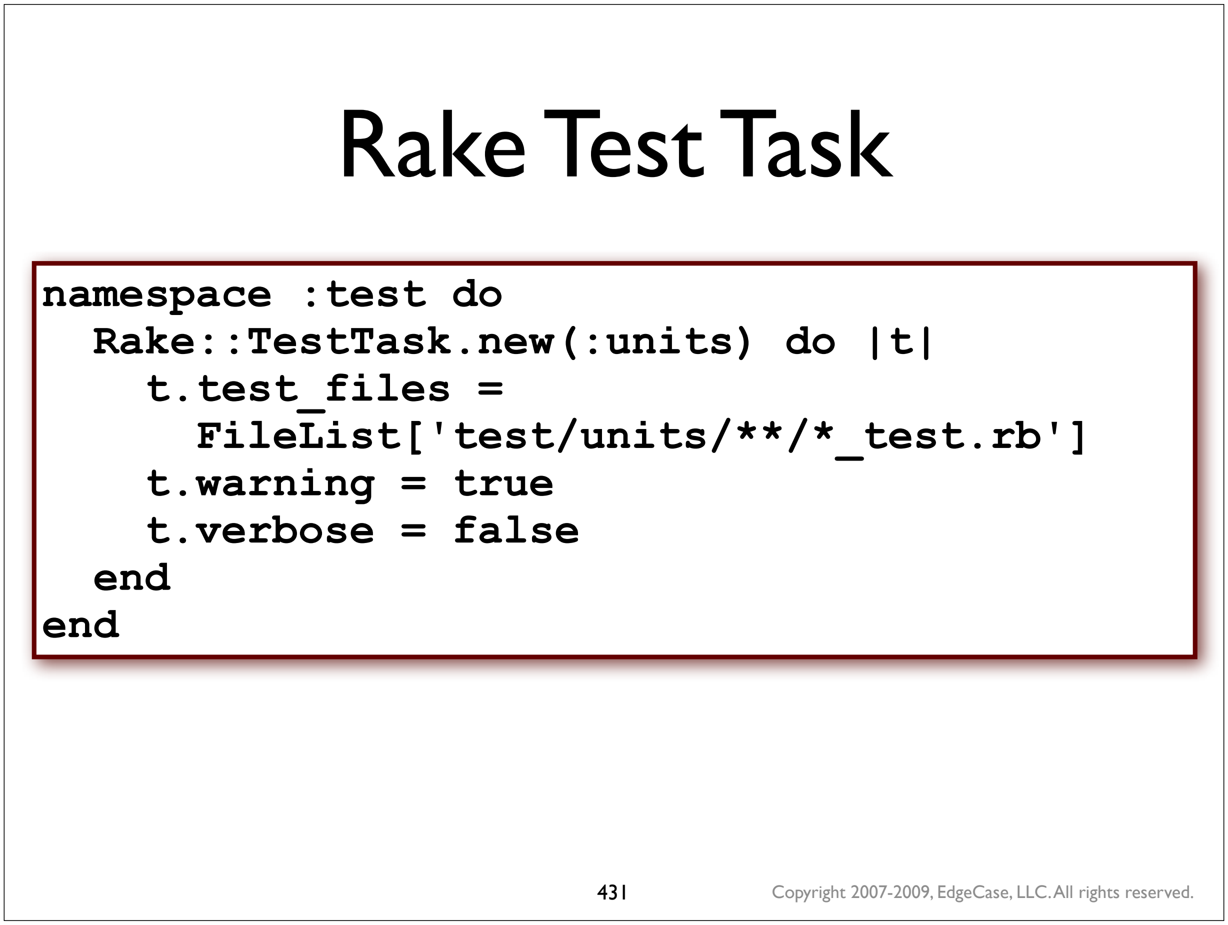
How well is your code tested?

Assumption:

If a line of code is never executed during a test, then the test cannot detect when that line is incorrect.

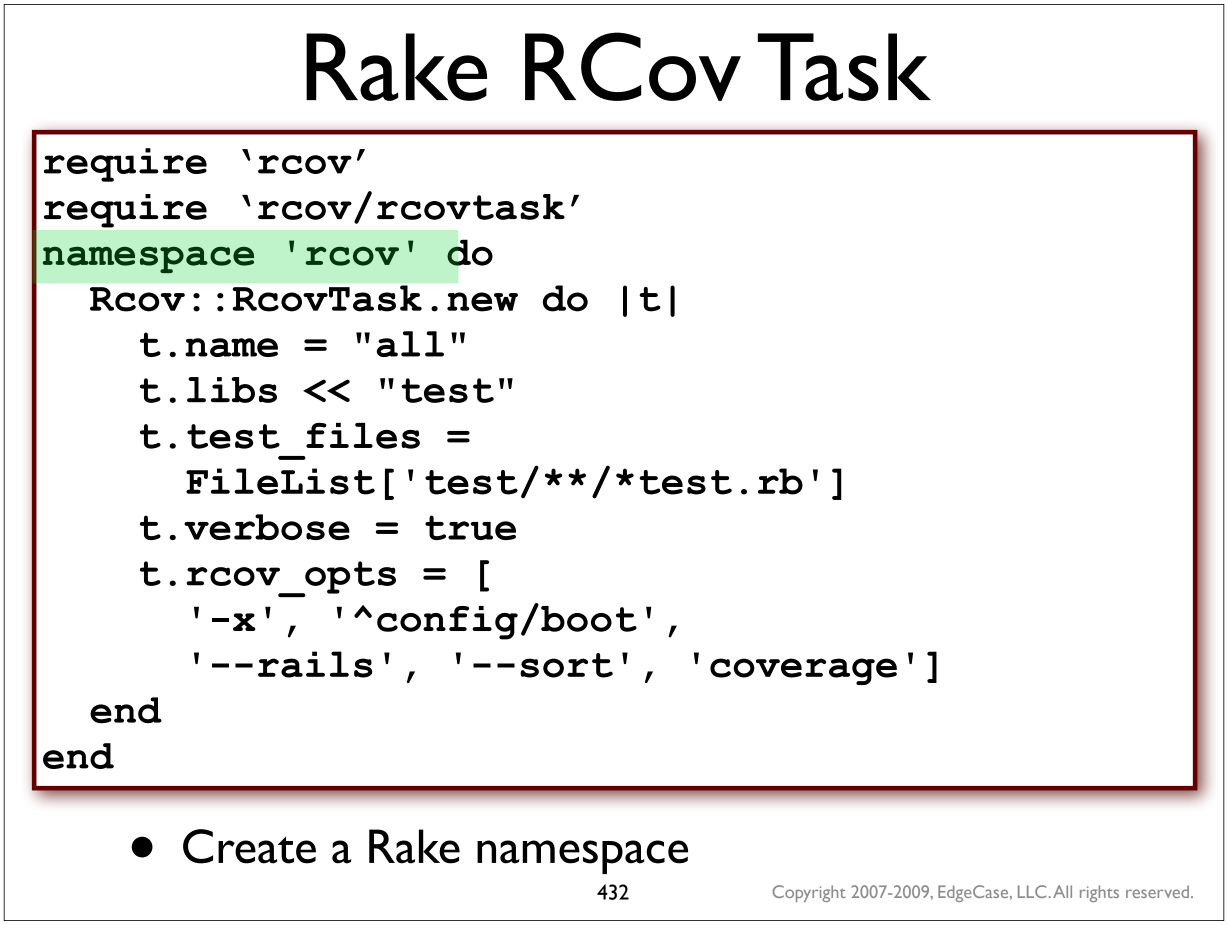
Need:

A Coverage Report Tool



Rake Test Task

```
namespace :test do
  Rake::TestTask.new(:units) do |t|
    t.test_files =
      FileList['test/units/**/*_test.rb']
    t.warning = true
    t.verbose = false
  end
end
```



Rake RCov Task

```
require ‘rcov’
require ‘rcov/rcovtask’
namespace 'rcov' do
  Rcov::RcovTask.new do |t|
    t.name = "all"
    t.libs << "test"
    t.test_files =
      FileList['test/**/*test.rb']
    t.verbose = true
    t.rcov_opts = [
      '-x', '^config/boot',
      '--rails', '--sort', 'coverage']
  end
end
```

- Create a Rake namespace

Rake RCov Task

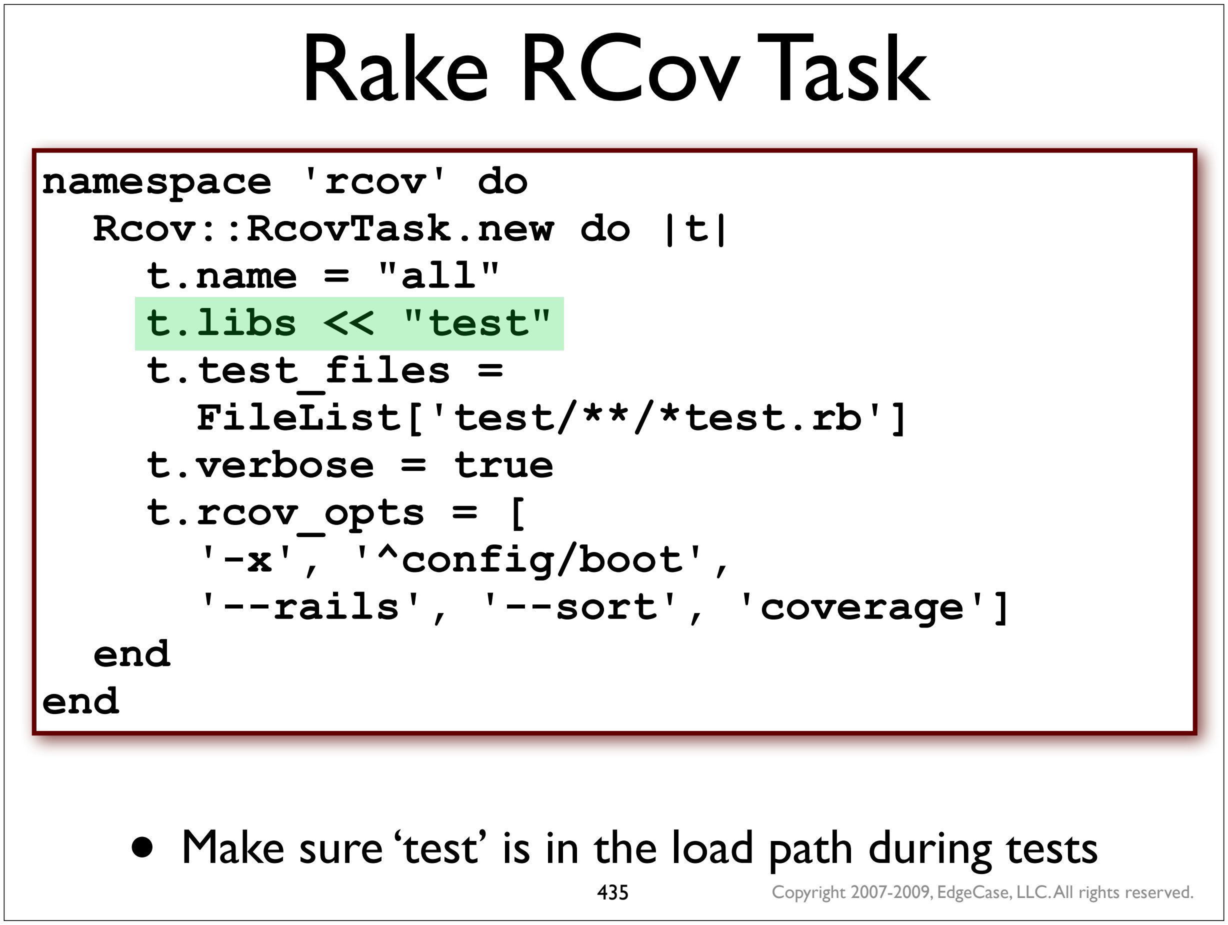
```
namespace 'rcov' do
  Rcov::RcovTask.new do |t|
    t.name = "all"
    t.libs << "test"
    t.test_files =
      FileList['test/**/*test.rb']
    t.verbose = true
    t.rcov_opts = [
      '-x', '^config/boot',
      '--rails', '--sort', 'coverage']
  end
end
```

- Handles the details of creating an RCov task with the following details

Rake RCov Task

```
namespace 'rcov' do
  Rcov::RcovTask.new do |t|
    t.name = "all"
    t.libs << "test"
    t.test_files =
      FileList['test/**/*test.rb']
    t.verbose = true
    t.rcov_opts = [
      '-x', '^config/boot',
      '--rails', '--sort', 'coverage']
  end
end
```

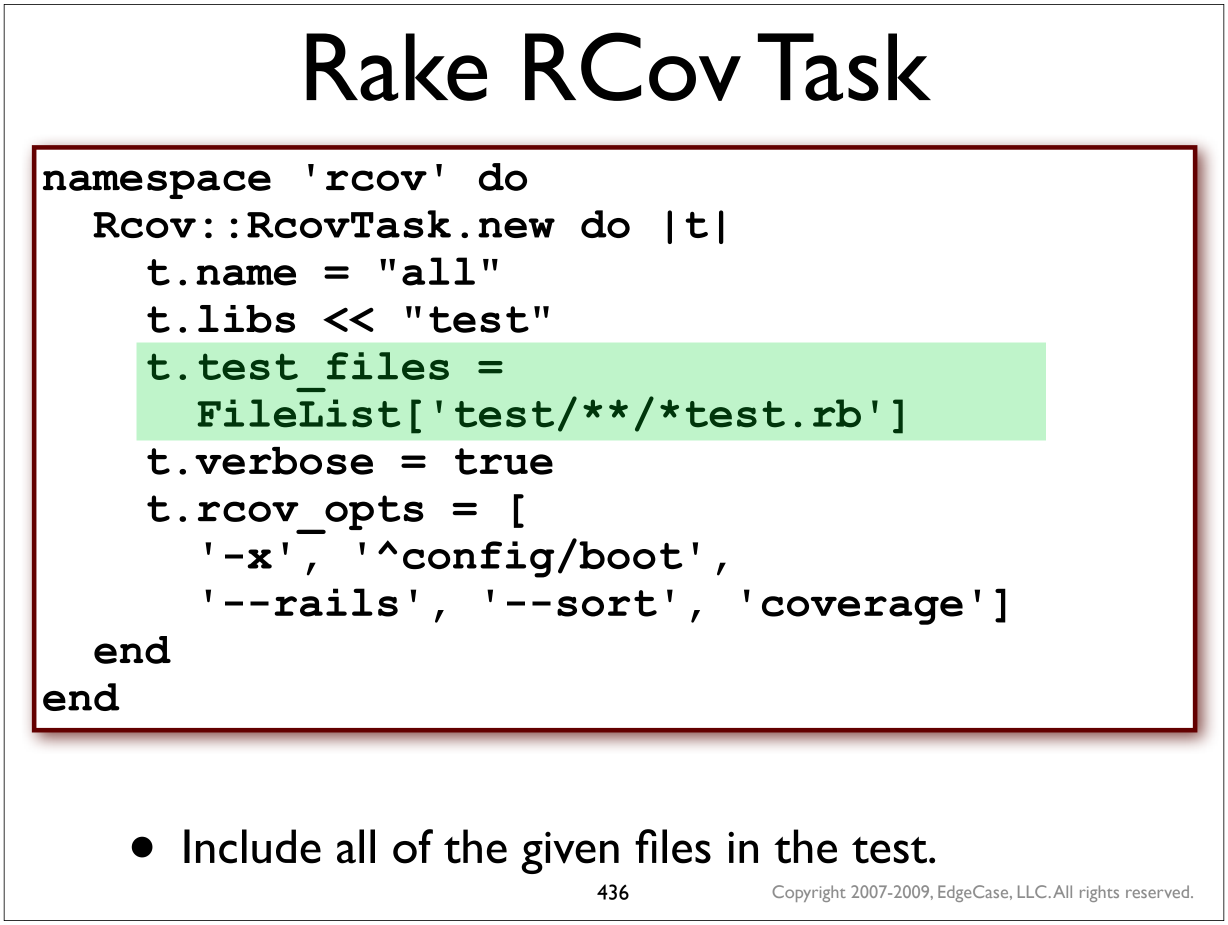
- Invoked via: rake rcov:units



Rake RCov Task

```
namespace 'rcov' do
  Rcov::RcovTask.new do |t|
    t.name = "all"
    t.libs << "test"
    t.test_files =
      FileList['test/**/*test.rb']
    t.verbose = true
    t.rcov_opts = [
      '-x', '^config/boot',
      '--rails', '--sort', 'coverage']
  end
end
```

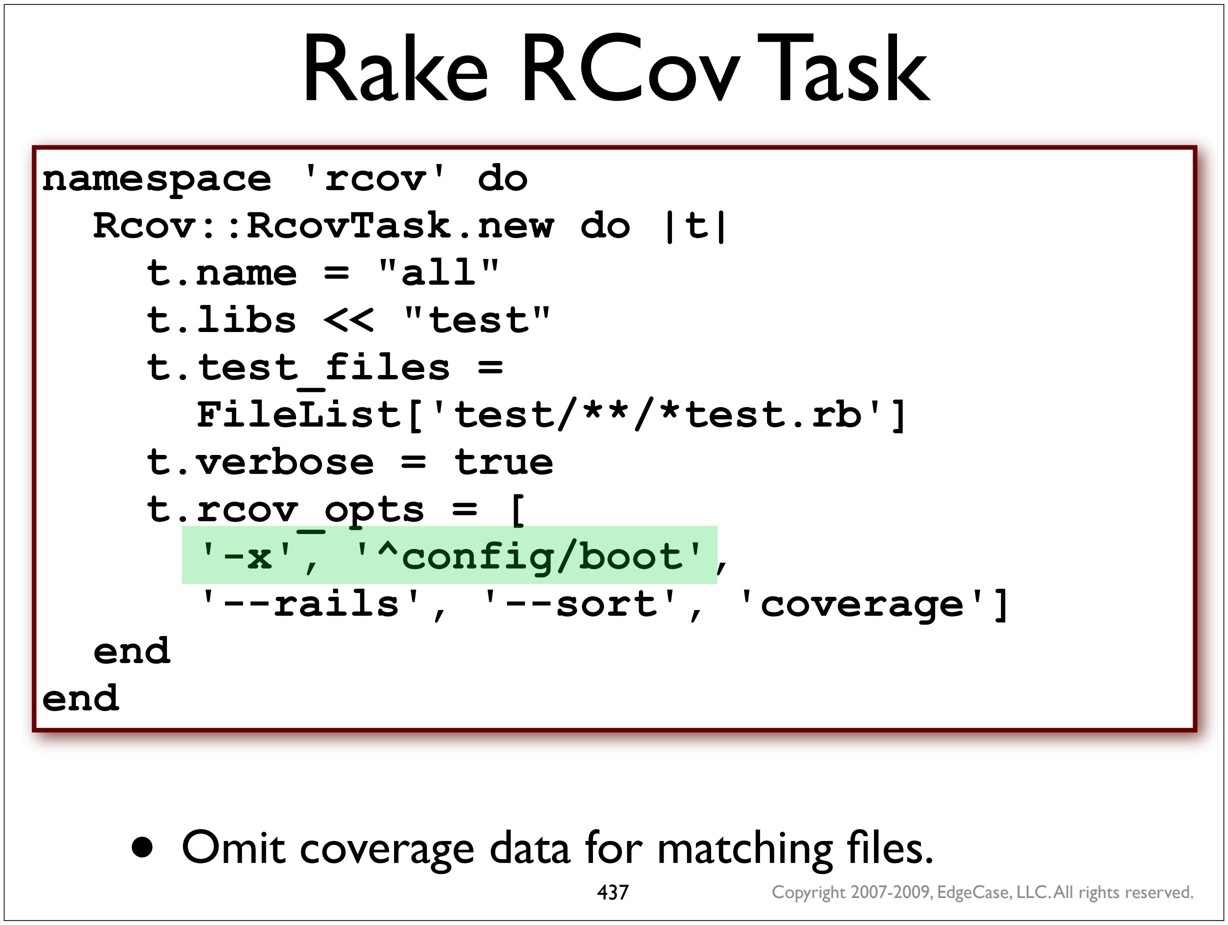
- Make sure ‘test’ is in the load path during tests



Rake RCov Task

```
namespace 'rcov' do
  Rcov::RcovTask.new do |t|
    t.name = "all"
    t.libs << "test"
    t.test_files =
      FileList['test/**/*test.rb']
    t.verbose = true
    t.rcov_opts = [
      '-x', '^config/boot',
      '--rails', '--sort', 'coverage']
  end
end
```

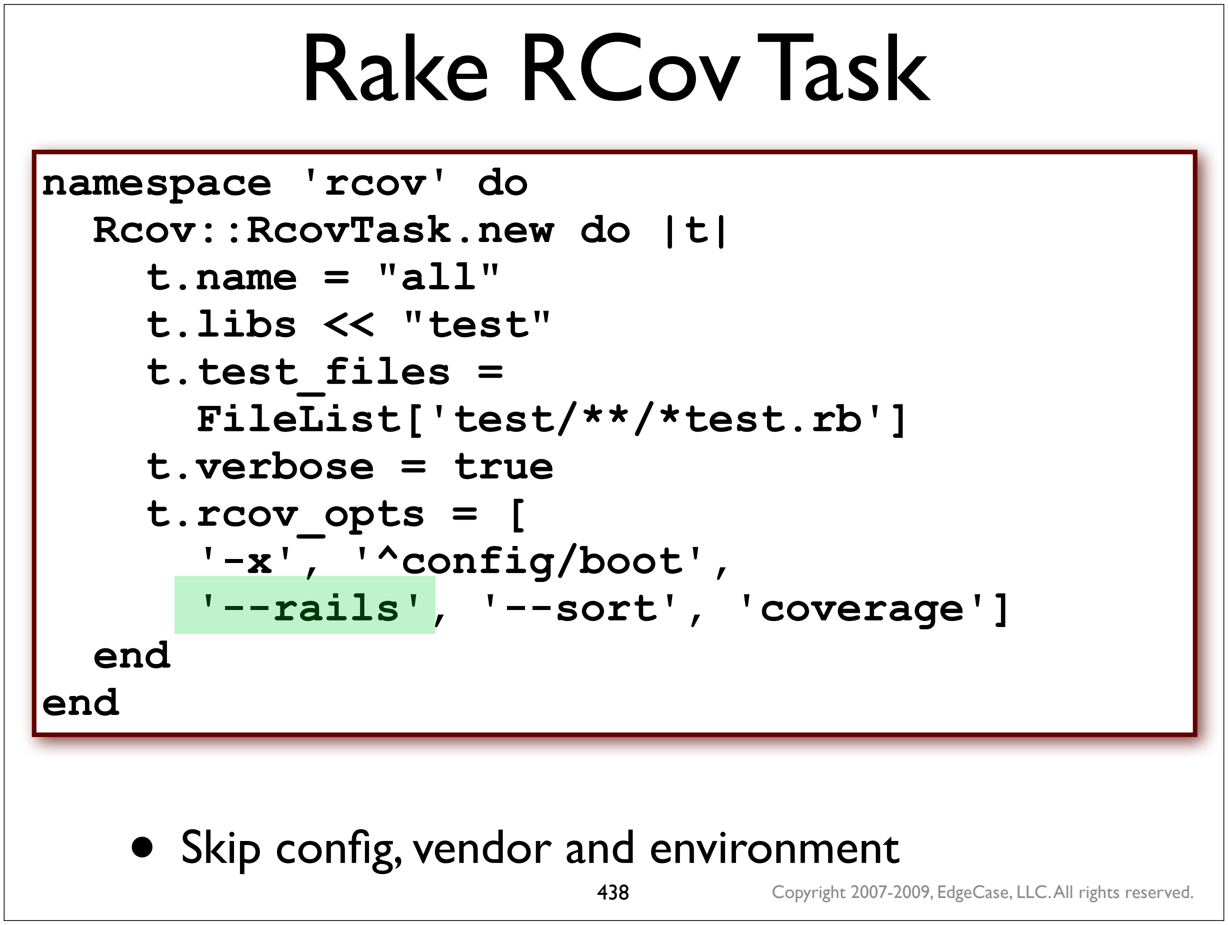
- Include all of the given files in the test.



Rake RCov Task

```
namespace 'rcov' do
  Rcov::RcovTask.new do |t|
    t.name = "all"
    t.libs << "test"
    t.test_files =
      FileList['test/**/*test.rb']
    t.verbose = true
    t.rcov_opts = [
      '-x', '^config/boot',
      '--rails', '--sort', 'coverage']
  end
end
```

- Omit coverage data for matching files.



Rake RCov Task

```
namespace 'rcov' do
  Rcov::RcovTask.new do |t|
    t.name = "all"
    t.libs << "test"
    t.test_files =
      FileList['test/**/*test.rb']
    t.verbose = true
    t.rcov_opts = [
      '-x', '^config/boot',
      '--rails', '--sort', 'coverage']
  end
end
```

- Skip config, vendor and environment

Rake RCov Task

```
namespace 'rcov' do
  Rcov::RcovTask.new do |t|
    t.name = "all"
    t.libs << "test"
    t.test_files =
      FileList['test/**/*test.rb']
    t.verbose = true
    t.rcov_opts = [
      '-x', '^config/boot',
      '--rails', '--sort', 'coverage']
  end
end
```

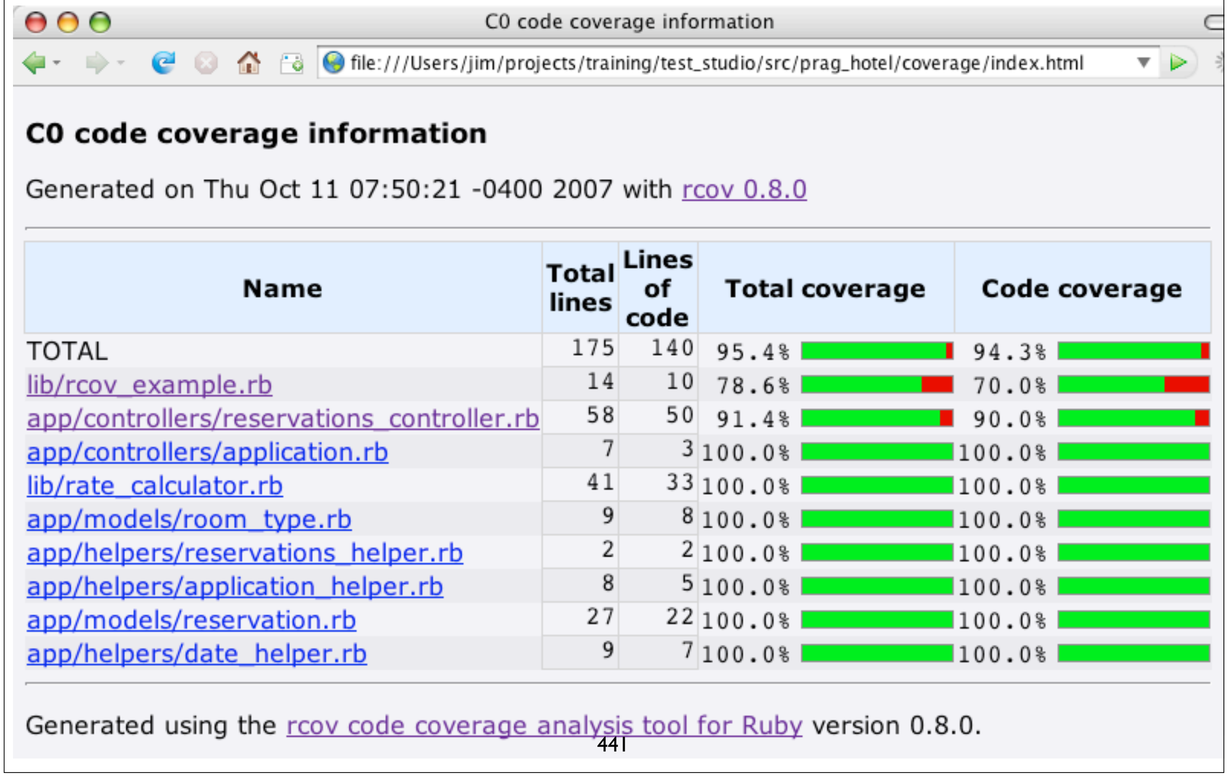
- Sort data by coverage, loc, or name.

Invoking RCov

```
$ rake rcov:all
(in /Users/jim/projects/training/test_studio/src/prag_hotel)
rm -r coverage
/opt/local/bin/ruby -Ilib:test -S rcov -x ^config/boot --rails
--sort coverage -o "coverage" "test/functional/
reservations_controller_test.rb" "test/unit/
rate_calculator_test.rb" "test/unit/rcov_example_test.rb" "test/
unit/reservation_test.rb" "test/unit/room_type_final_test.rb"
"test/unit/room_type_test.rb" "test/unit/helpers/
application_helper_test.rb" "test/unit/helpers/
date_helper_test.rb" "test/views/reservations_view_test.rb"
Loaded suite /opt/local/bin/rcov
Started
....................................
Finished in 0.978719 seconds.

36 tests, 117 assertions, 0 failures, 0 errors
$
```

- Looks like a test run.

open coverage/index.html



C0 code coverage information
file:///Users/jim/projects/training/test_studio/src/prag_hotel/coverage/index.html

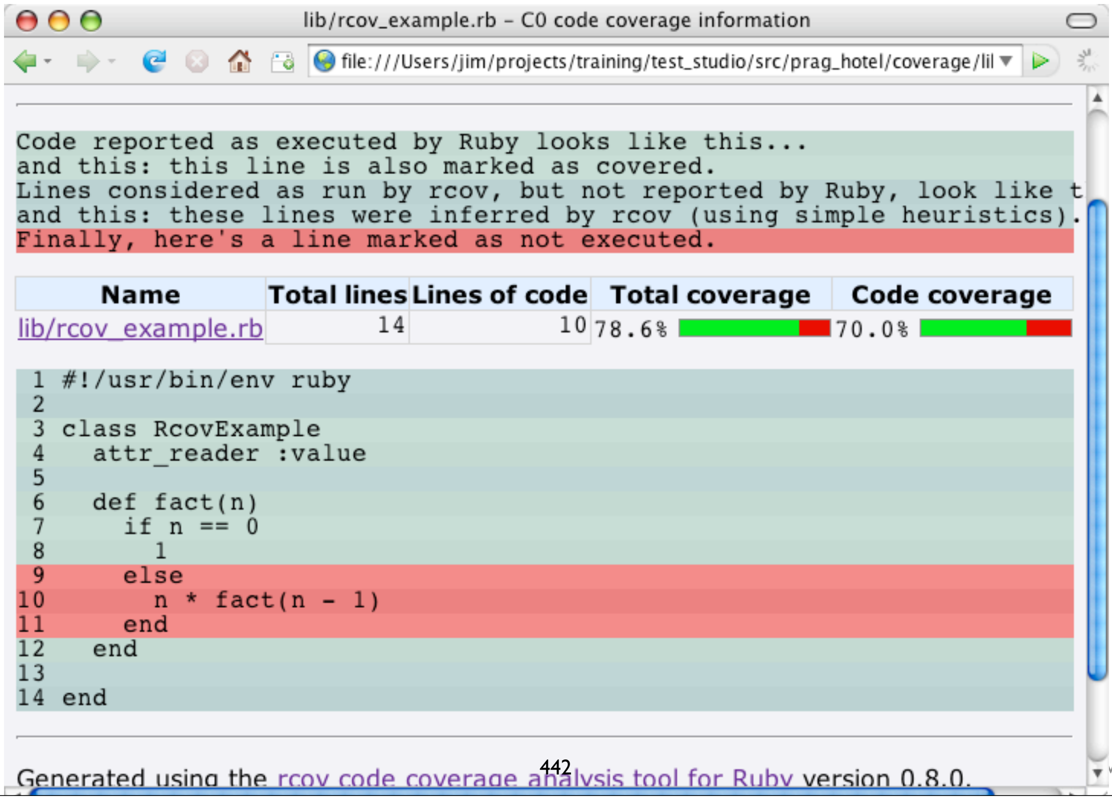
C0 code coverage information

Generated on Thu Oct 11 07:50:21 -0400 2007 with rcov 0.8.0

Name	Total lines	Lines of code	Total coverage	Code coverage
TOTAL	175	140	95.4%	94.3%
lib/rcov_example.rb	14	10	78.6%	70.0%
app/controllers/reservations_controller.rb	58	50	91.4%	90.0%
app/controllers/application.rb	7	3	100.0%	100.0%
lib/rate_calculator.rb	41	33	100.0%	100.0%
app/models/room_type.rb	9	8	100.0%	100.0%
app/helpers/reservations_helper.rb	2	2	100.0%	100.0%
app/helpers/application_helper.rb	8	5	100.0%	100.0%
app/models/reservation.rb	27	22	100.0%	100.0%
app/helpers/date_helper.rb	9	7	100.0%	100.0%

Generated using the rcov code coverage analysis tool for Ruby version 0.8.0.

RCov Detail



lib/rcov_example.rb - C0 code coverage information
file:///Users/jim/projects/training/test_studio/src/prag_hotel/coverage/lil

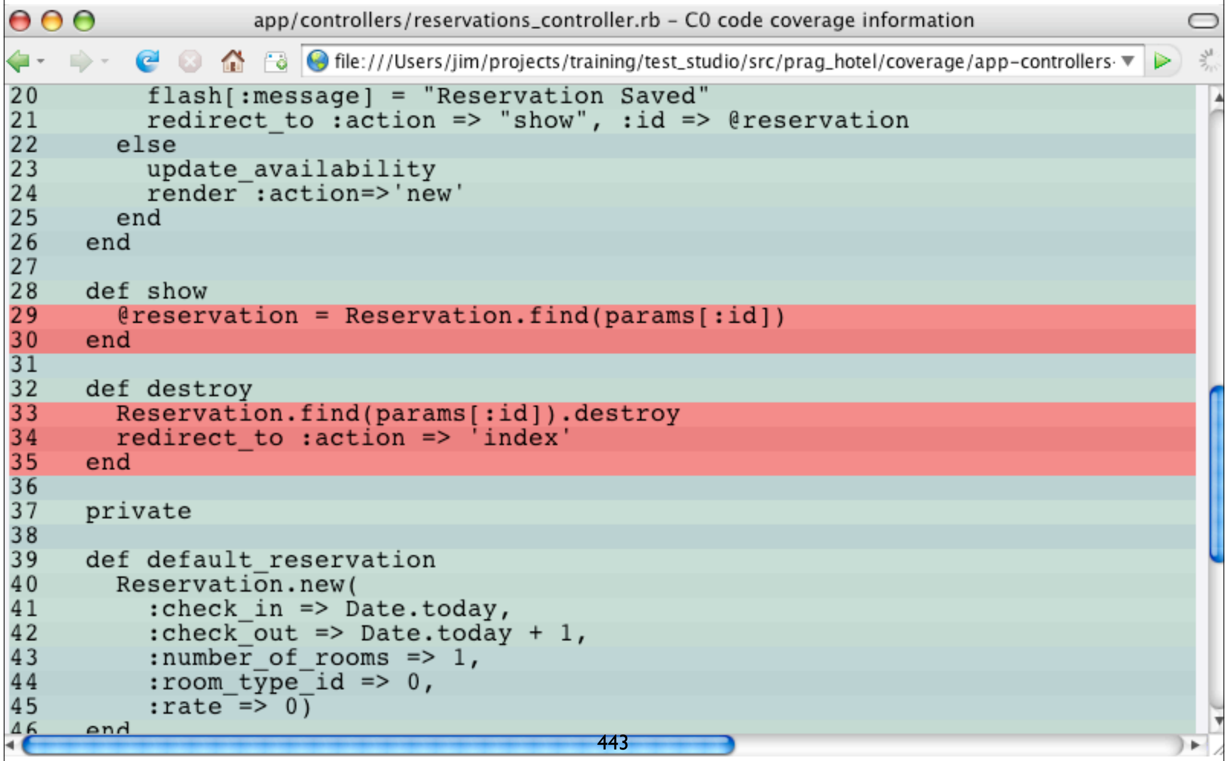
```
Code reported as executed by Ruby looks like this...
and this: this line is also marked as covered.
Lines considered as run by rcov, but not reported by Ruby, look like t
and this: these lines were inferred by rcov (using simple heuristics).
Finally, here's a line marked as not executed.
```

Name	Total lines	Lines of code	Total coverage	Code coverage
lib/rcov_example.rb	14	10	78.6%	70.0%

```
#!/usr/bin/env ruby

class RcovExample
  attr_reader :value

  def fact(n)
    if n == 0
      1
    else
      n * fact(n - 1)
    end
  end

end
```

Generated using the rcov code coverage analysis tool for Ruby version 0.8.0.

RCov Detail



app/controllers/reservations_controller.rb – C0 code coverage information
file:///Users/jim/projects/training/test_studio/src/prag_hotel/coverage/app-controllers-

```
      flash[:message] = "Reservation Saved"
      redirect_to :action => "show", :id => @reservation
    else
      update_availability
      render :action=>'new'
    end
  end

  def show
    @reservation = Reservation.find(params[:id])
  end

  def destroy
    Reservation.find(params[:id]).destroy
    redirect_to :action => 'index'
  end

  private

  def default_reservation
    Reservation.new(
      :check_in => Date.today,
      :check_out => Date.today + 1,
      :number_of_rooms => 1,
      :room_type_id => 0,
      :rate => 0)
  end
```

RCov Tricks: Evals

```
if method_name.to_s =~ /=$/

  sclass.class_eval %{
    def #{method_name}(*args, &block)
      @flexmock_proxy.mock.
        __send__(:#{method_name}, *args, &block)
    end
  }
else
...
```

- RCov never sees this code as covered

RCov Tricks: Evals

```
if method_name.to_s =~ /=$/
  eval_line = __LINE__ + 1
  sclass.class_eval %{
    def #{method_name}(*args, &block)
      @flexmock_proxy.mock.
        __send__(:#{method_name}, *args, &block)
    end
  }, __FILE__, eval_line
else
...
```

- Add file and line arguments to eval

RCov Tricks: Evals

```
def define_proxy_method(method_name)
  if method_name.to_s =~ /=$/
    eval_line = __LINE__ + 1
    sclass.class_eval %{
      def #{method_name}(*args, &block)
        @flexmock_proxy.mock.__send__(:#{method_name}, *args, &block)
      end
    }, __FILE__, eval_line
  else
    eval_line = __LINE__ + 1
    sclass.class_eval %{
      def #{method_name}(*args, &block)
        @flexmock_proxy.mock.#{method_name}(*args, &block)
      end
    }, __FILE__, eval_line
    make_rcov_recognize_the_above_eval_is_covered = true
  end
end
```

- Sometimes a little more *encouragement* is required

What is a good code coverage number?

80% ?

90% ?

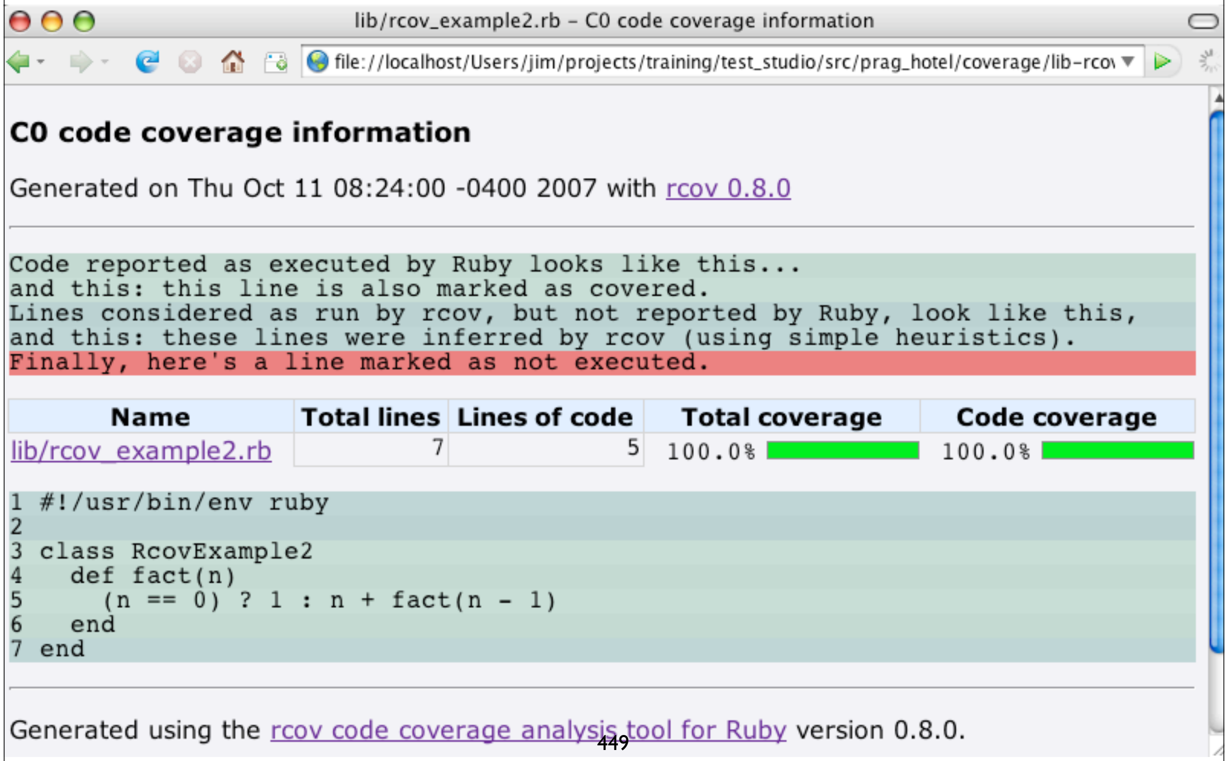
100% ?

If you have 100% Code Coverage ...

Does that imply that your code is well tested?

NO

Well Covered?



lib/rcov_example2.rb - C0 code coverage information

file://localhost/Users/jim/projects/training/test_studio/src/prag_hotel/coverage/lib-rco\

C0 code coverage information

Generated on Thu Oct 11 08:24:00 -0400 2007 with rcov 0.8.0

```
Code reported as executed by Ruby looks like this...
and this: this line is also marked as covered.
Lines considered as run by rcov, but not reported by Ruby, look like this,
and this: these lines were inferred by rcov (using simple heuristics).
Finally, here's a line marked as not executed.
```

Name	Total lines	Lines of code	Total coverage	Code coverage
lib/rcov_example2.rb	7	5	100.0% 🟩	100.0% 🟩

```
#!/usr/bin/env ruby

class RcovExample2
  def fact(n)
    (n == 0) ? 1 : n + fact(n - 1)
  end
end
```

Generated using the rcov code coverage analysis tool for Ruby version 0.8.0.

Beware of:

- Trinary Operators
- Complex Conditionals
  - Operators: | |, &&
  - Keywords: and, or

The above can hide code from a C0 coverage tool (like RCov)

Line VS Path Coverage

- RCov provides C0 code coverage
  - i.e. Lines of code are counted
- C1 code == Path coverage
  - i.e. Every path through your code is counted

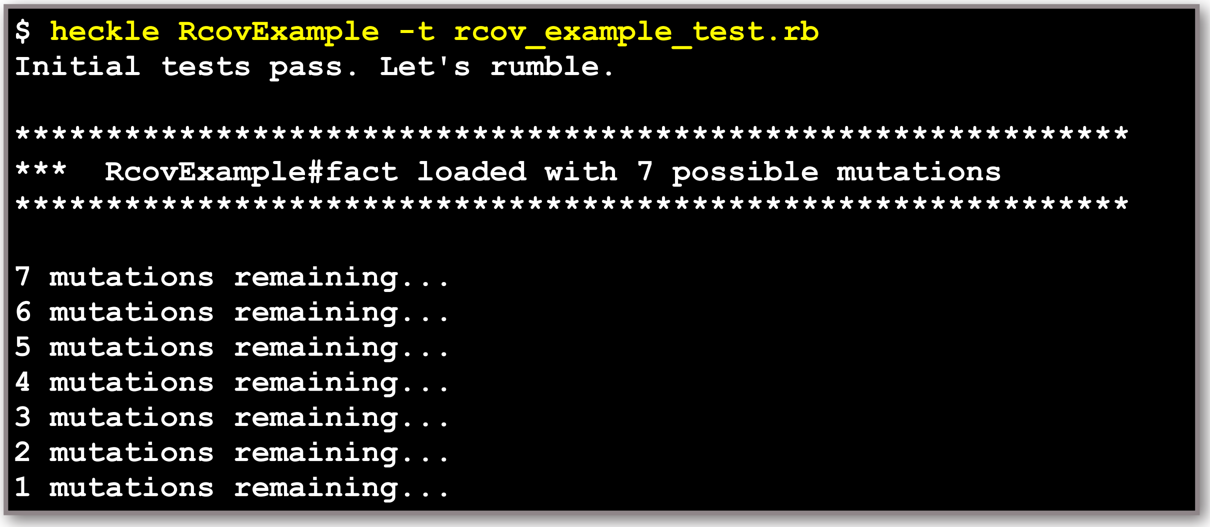
Code with 100% C0 Coverage

```
class RcovExample
  def fact(n)
    (n == 0) ? 1 : n * fact(n - 1)
  end
end
```

- What would happen of we changed “n - 1” in the above code to “n - 2”.
  - Should the tests break?
  - What does it mean of the tests **don’t** break?

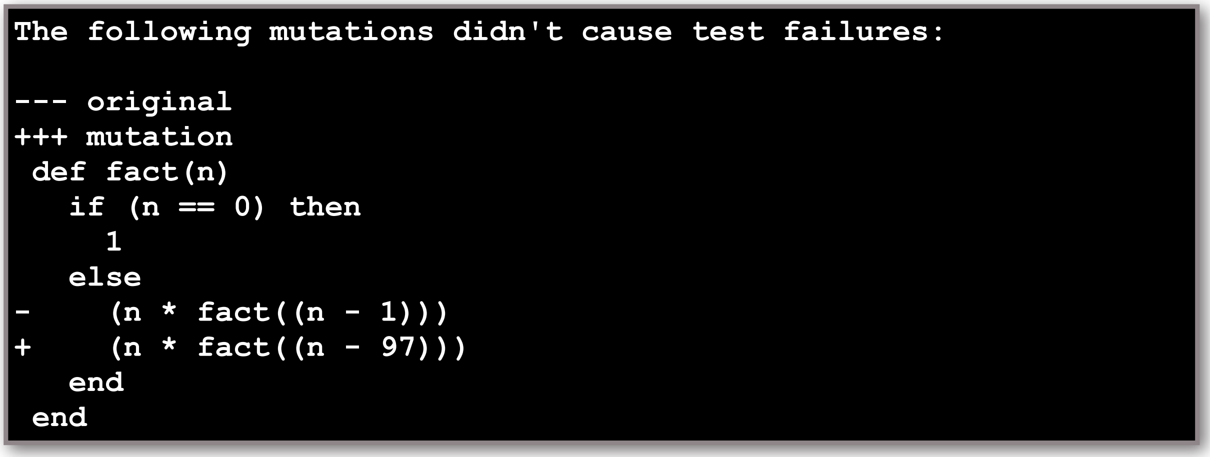
Heckle

Running Heckle



```
$ heckle RcovExample -t rcov_example_test.rb
Initial tests pass. Let's rumble.

*************************************************************
***  RcovExample#fact loaded with 7 possible mutations
*************************************************************

7 mutations remaining...
6 mutations remaining...
5 mutations remaining...
4 mutations remaining...
3 mutations remaining...
2 mutations remaining...
1 mutations remaining...
```

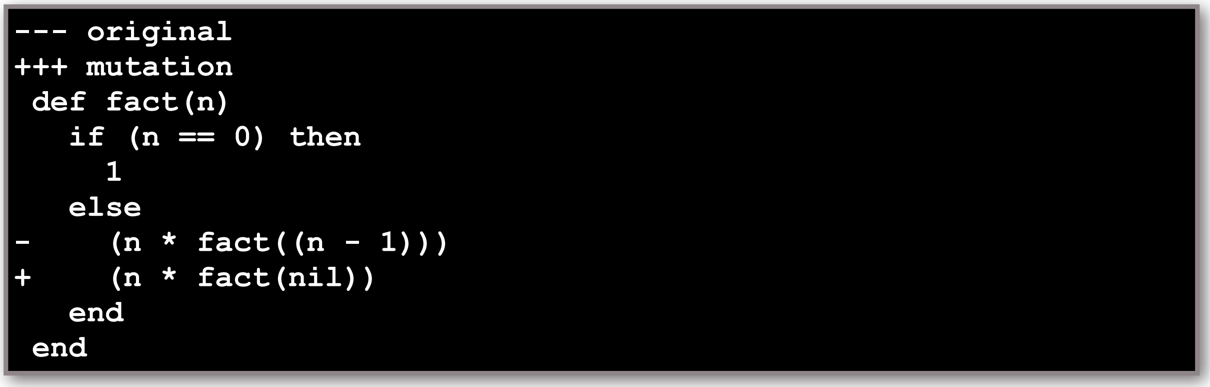
Running Heckle



```
The following mutations didn't cause test failures:

--- original
+++ mutation
 def fact(n)
   if (n == 0) then
     1
   else
-    (n * fact((n - 1)))
+    (n * fact((n - 97)))
   end
 end
```

- Changed “n-1” to “n-97” and the test did **not** fail

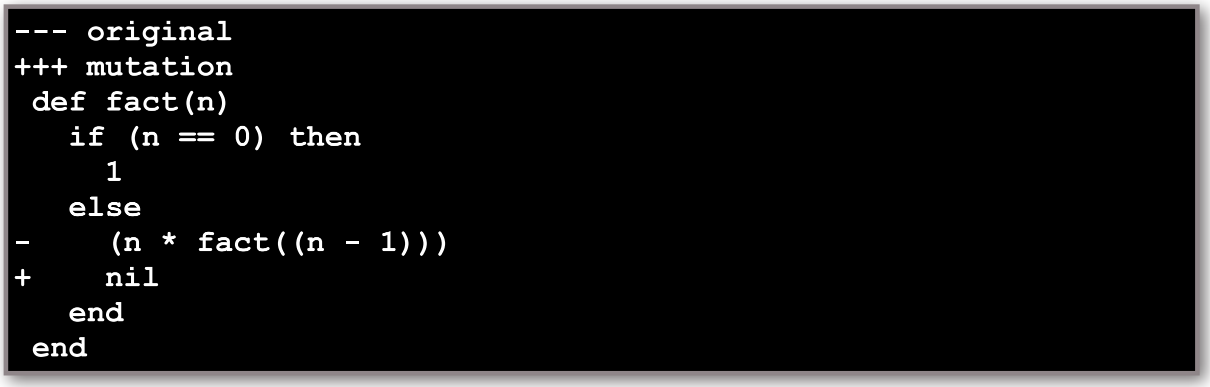
Running Heckle



```
--- original
+++ mutation
 def fact(n)
   if (n == 0) then
     1
   else
-    (n * fact((n - 1)))
+    (n * fact(nil))
   end
 end
```

- Changed “n-1” to “nil” and the test did **not** fail

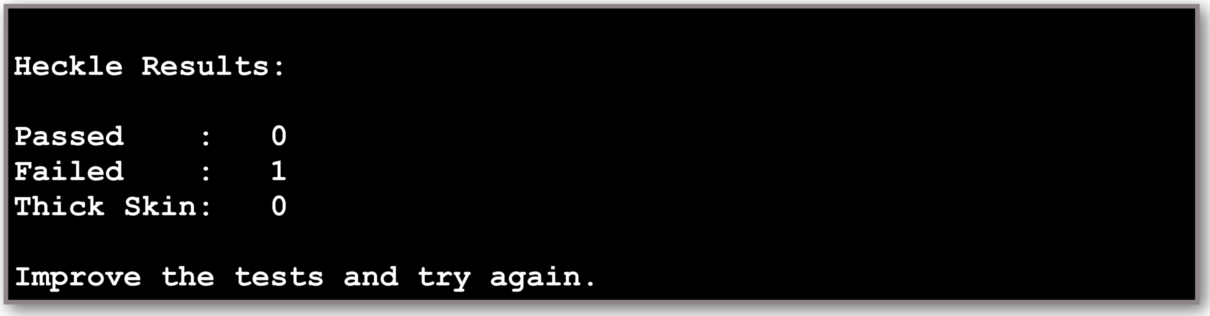
Running Heckle



```
--- original
+++ mutation
 def fact(n)
   if (n == 0) then
     1
   else
-    (n * fact((n - 1)))
+    nil
   end
 end
```

- Changed the return value on the else branch to “nil” and the test did **not** fail

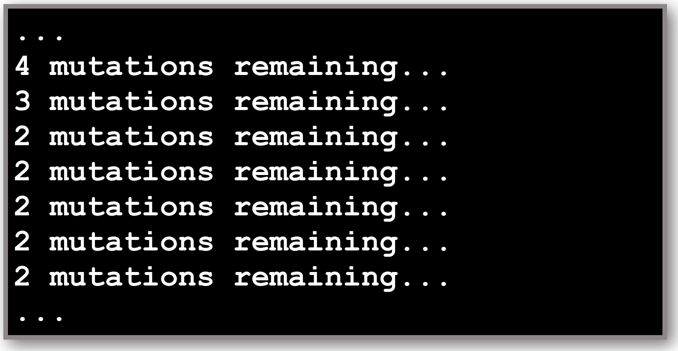
Running Heckle



```
Heckle Results:

Passed    :   0
Failed    :   1
Thick Skin:   0

Improve the tests and try again.
```

Heckle Gotchas

- All your tests must pass before Heckle is run.
- Sometimes you get stuck on 2 mutations



```
...
4 mutations remaining...
3 mutations remaining...
2 mutations remaining...
2 mutations remaining...
2 mutations remaining...
2 mutations remaining...
2 mutations remaining...
...
```

Mutations?

- Replace constants and expressions with different values
- Change “condition” to “!condition” in ifs and whiles
- Reverse if/else branches

What does Heckle tell you about your code?

Heckle Tells You ...

- If Heckle fails your code:
  - Then your tests are too anemic to cover all of the possible paths through your code base.

There are two kinds of code that Heckle smiles upon...

- Clean code that is well tested
- Convoluted spaghetti code that is so twisted that changing anything breaks everything.

What can tell you that you have good code and tests?

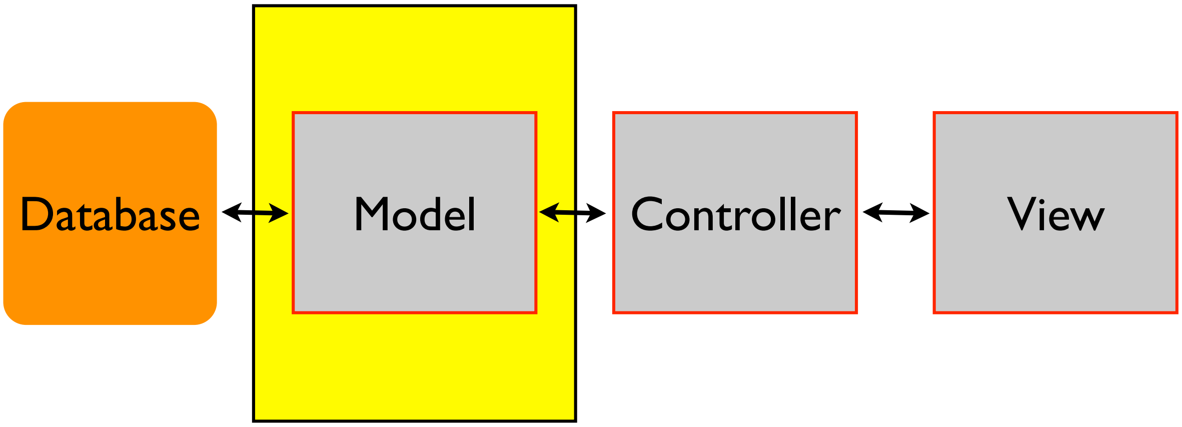
- Test Driven Design
- Pair Programming
- Continuous Integration
- Test-Infected Developers

Lab 8:
Increase Coverage

Integration Testing

our focus so far ...

Unit Testing



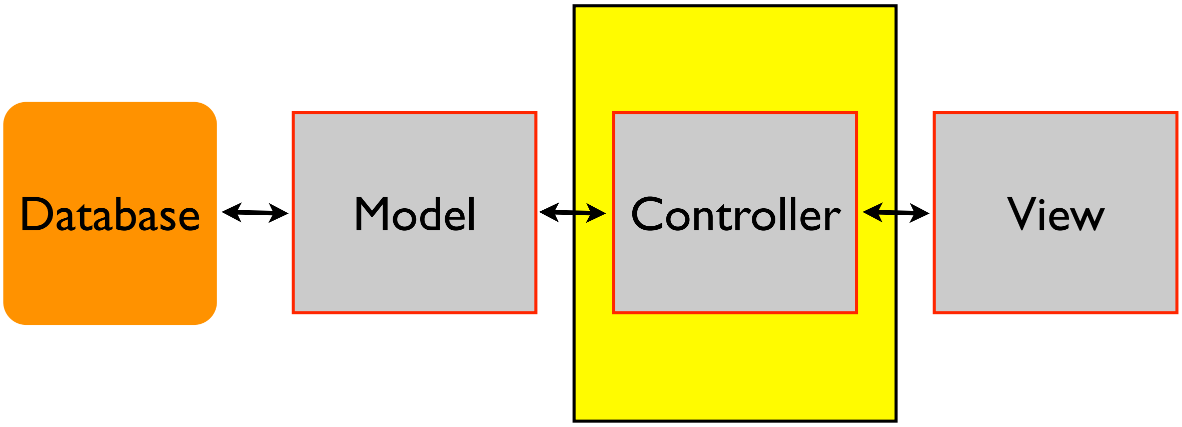
Database

Model

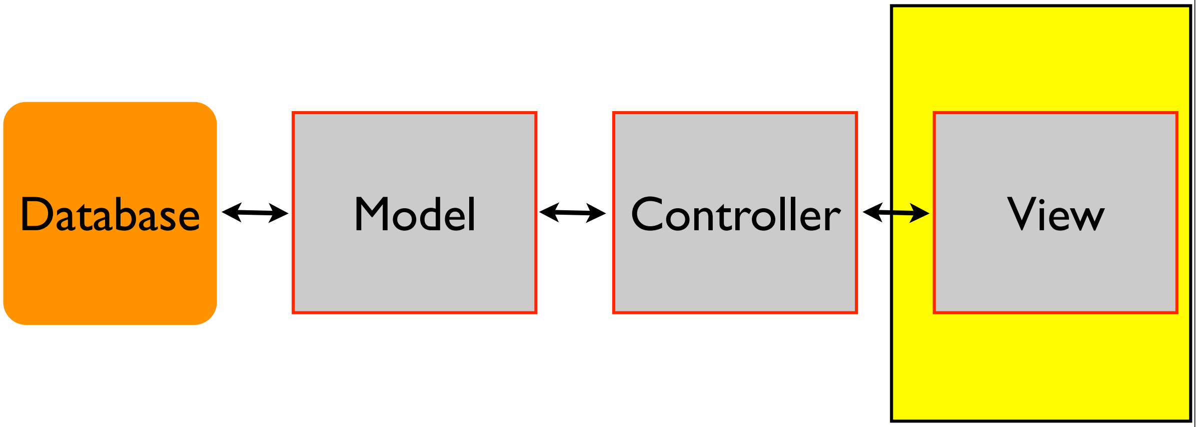
Controller

View

Unit Testing

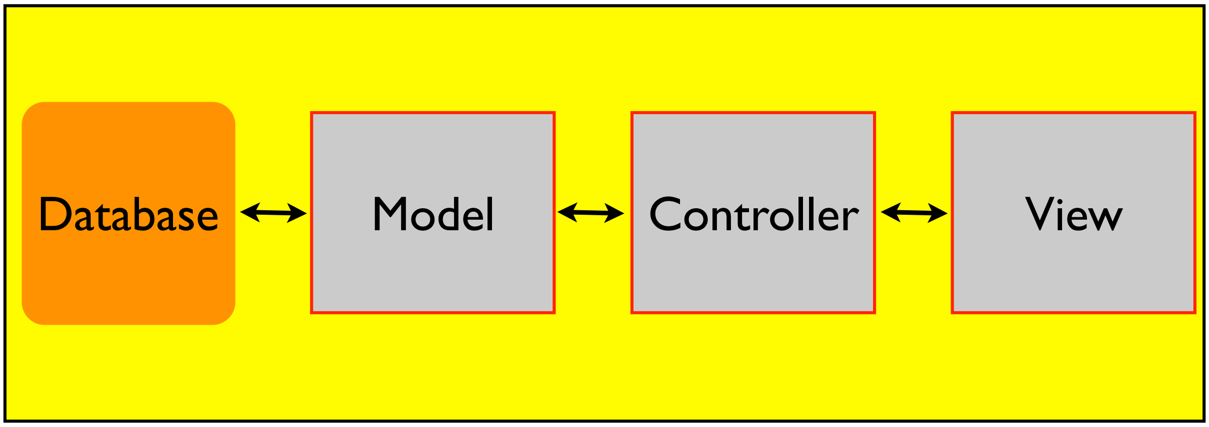


Unit Testing



Now it’s time to ...

Test all the parts together



Database ↔ Model ↔ Controller ↔ View

Unit Testing Provides

- Confidence to change and refactor
- Safety net of the parts
- Documentation of intent and design

Unit Testing does not ensure

- ... it all works together
- ... the whole system is correct
- ... it keeps working together

4 Types

- Rails Integration Tests
- Selenium in-browser testing
- Watir Testing
  - Watir
  - SafariWatir
  - FireWatir
- Cucumber

Rails Integration

Rails Integration Tests

- Tests your app outside of the browser
- Tests the orchestration of your app
- Can flow along several cycles

test/integration

```
require File.dirname(__FILE__) + '/../test_helper'

class MakeReservationTest <
              ActionController::IntegrationTest
  fixtures :reservations, :room_types

  ...

end
```

```
def test_creating_a_reservation
    count = Reservation.count

    get "/reservations/new", :reservation => {
        :name => 'Jones', :number_of_rooms => 3,
        :check_in => '2007-05-05', :check_out => '2007-05-06' }

    assert_response :success
    assert_template 'new'

    post "/reservations/create",
      :reservation => Reservation.valid_options

    assert_response :redirect
    assert_equal count + 1, Reservation.count
end
```

demo

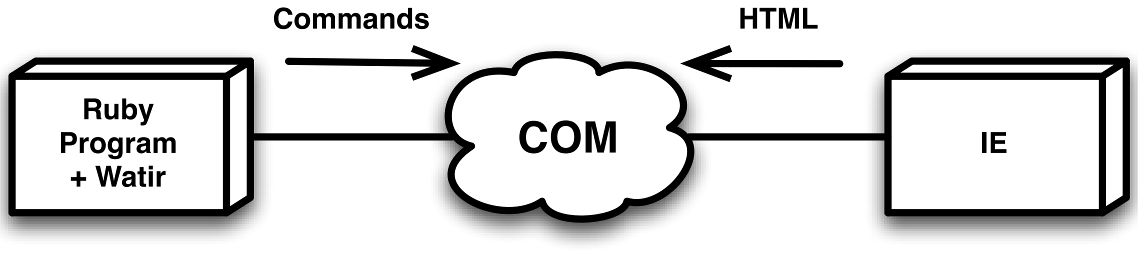
comes up a little short

- Still abstracts some steps
- Does not help in browsers
- does not test JavaScript / RJS

Watir
FireWatir
SafariWatir

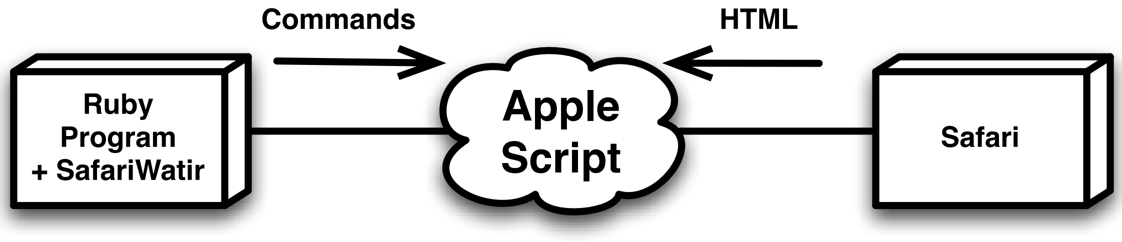
Test Driven Development with Rails

Original Watir



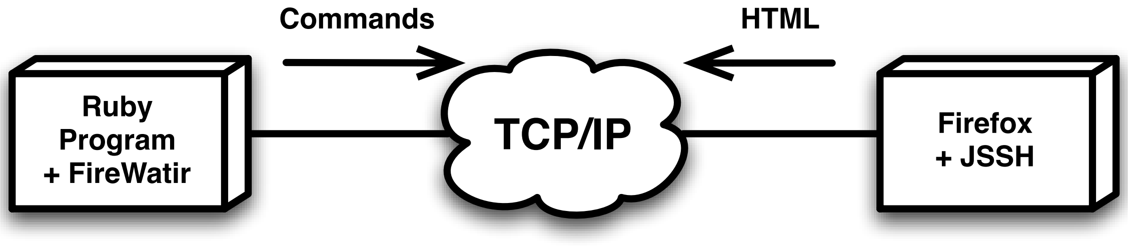
- Windows & IE only

SafariWatir



- Port of Watir to Mac
- Uses AppleScript to communicate to Safari

FireWatir



- FireFox Based
- Requires JSSH plugin to talk to browser

Installing
Watir /SafariWatir

- gem install watir -y
- gem install safariwatir -y

Installing FireWatir

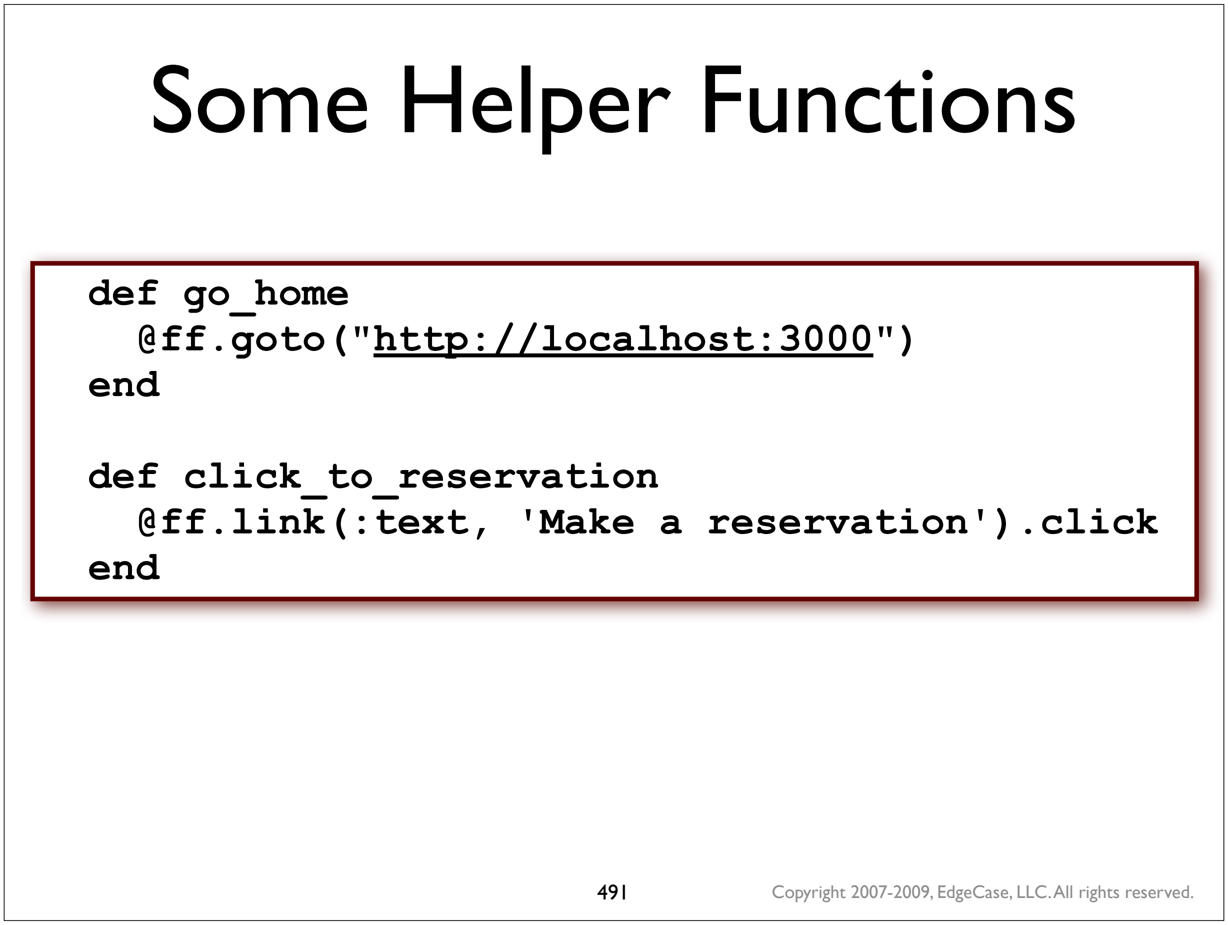
- Install Firefox 1.5 or higher
- Download the JSSH firefox extension
  - http://code.google.com/p/firewatir/
- From Firefox, open the JSSH .xpi file and install it.
- Close Firefox
- Start Firefox from the command line with:
  - firefox-bin -jssh
- Telnet to localhost 9997 to check for working jssh
- gem install firewatir -y

Try it in IRB

```
$ irb
>> require 'firewatir'
=> true
>> ff = FireWatir::Firefox.new
=> #<FireWatir::Firefox:0x139c604 @window_url="http://
localhost:3000/reservations/new", @window_title="">
>> ff.goto("http://localhost:3000")
=> "http://localhost:3000/"
>> ff.link(:text, "Make a reservation").click
=> 0
>> ff.select_list(:id, "reservation_check_out_3i").value = 12
=> 12
>> ff.button(:value, "Change Requested Dates").click
=> 0
>> ff.button(:index, 1).click
=> 0
```

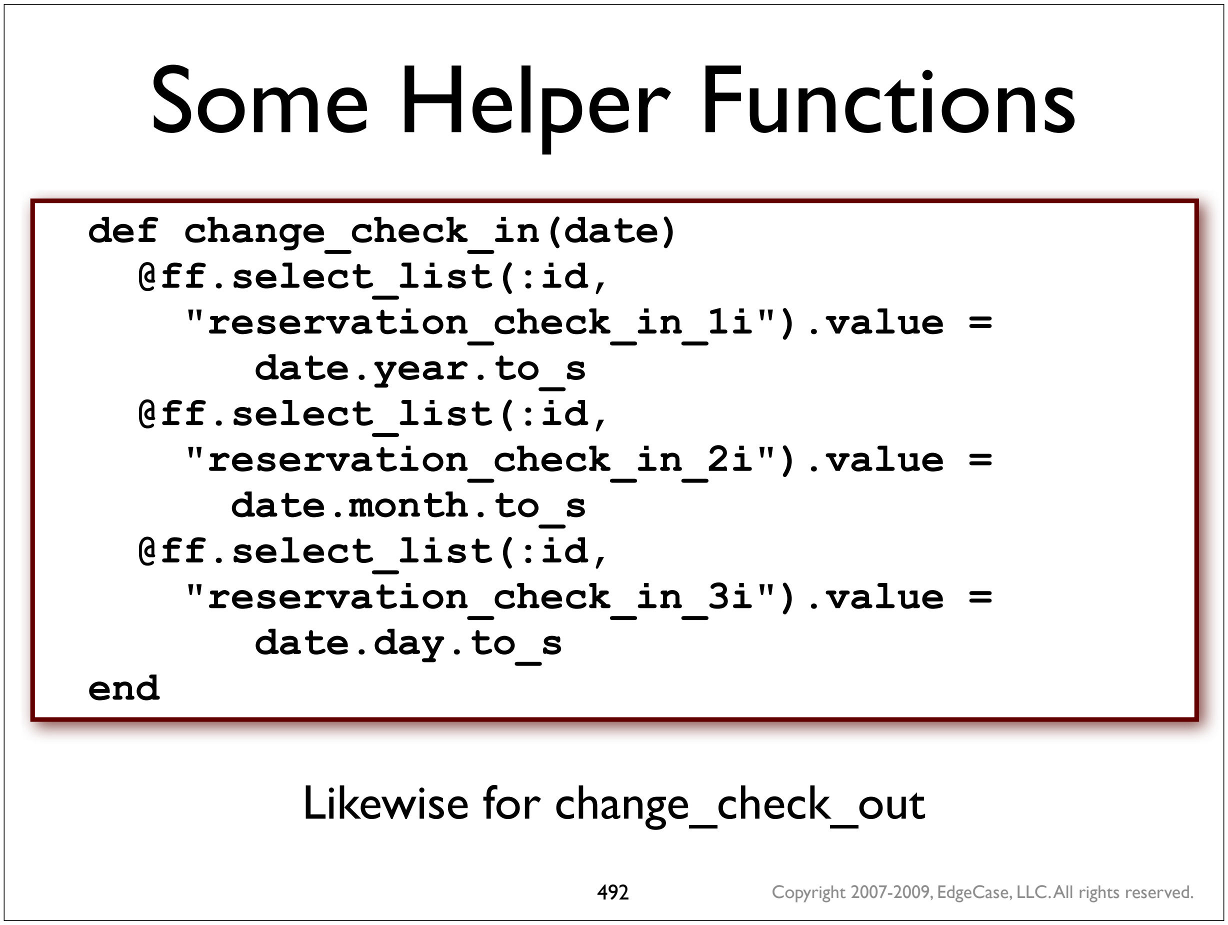
Testing with Watir

```
  def setup
    @ff = FireWatir::Firefox.new
  end
```



Some Helper Functions

```
def go_home
  @ff.goto("http://localhost:3000")
end

def click_to_reservation
  @ff.link(:text, 'Make a reservation').click
end
```



Some Helper Functions

```
def change_check_in(date)
  @ff.select_list(:id,
    "reservation_check_in_1i").value =
       date.year.to_s
  @ff.select_list(:id,
    "reservation_check_in_2i").value =
      date.month.to_s
  @ff.select_list(:id,
    "reservation_check_in_3i").value =
       date.day.to_s
end
```

Likewise for change_check_out

test_home_page

```
def test_home_page
  go_home
  assert_equal "Pragmatic Hotel", @ff.title
end
```

test_make_a_reservation

```
def test_make_a_reservation
  go_home
  click_to_reservation
  change_check_in(Date.new(2008, 2, 14))
  change_check_out(Date.new(2008, 2, 15))
  @ff.button(:value,
    "Change Requested Dates").click
  @ff.button(:index, 1).click

  assert_match(/Check In: 2008-02-14/,
    @ff.text)
  assert_match(/Check Out: 2008-02-15/,
    @ff.text)
end
```

Navigation Commands

Goto a URL reference

```
ff.goto(“http://google.com”)
```

Goto previous page

```
ff.back
```

Refresh the page

```
ff.refresh
```

Query the current URL

```
ff.url    # => Current URL
```

Query Commands

Page Title

```
ff.title
```

Get the text of a page

```
ff.text
```

Document object for page

```
ff.document
```

The URL of the current page

```
ff.url
```

Show Commands

```
ff.show_all_objects
ff.show_divs
ff.show_forms
ff.show_frames
ff.show_images
ff.show_labels
ff.show_links
ff.show_pres
ff.show_spans
ff.show_tables
```

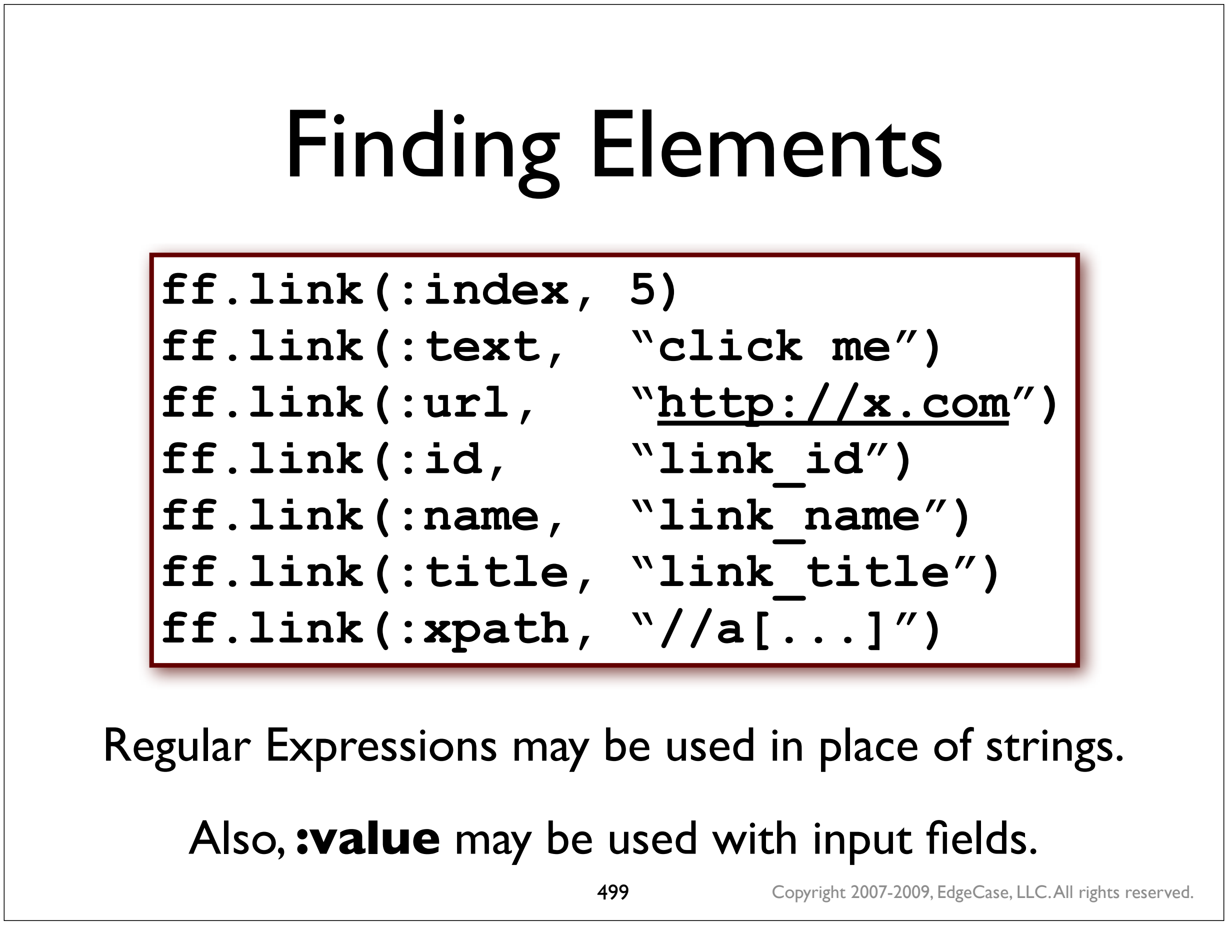
Elements

Find a single element of a type

```
ff.link(key, value)
```

Find all elements of a type

```
ff.links
```

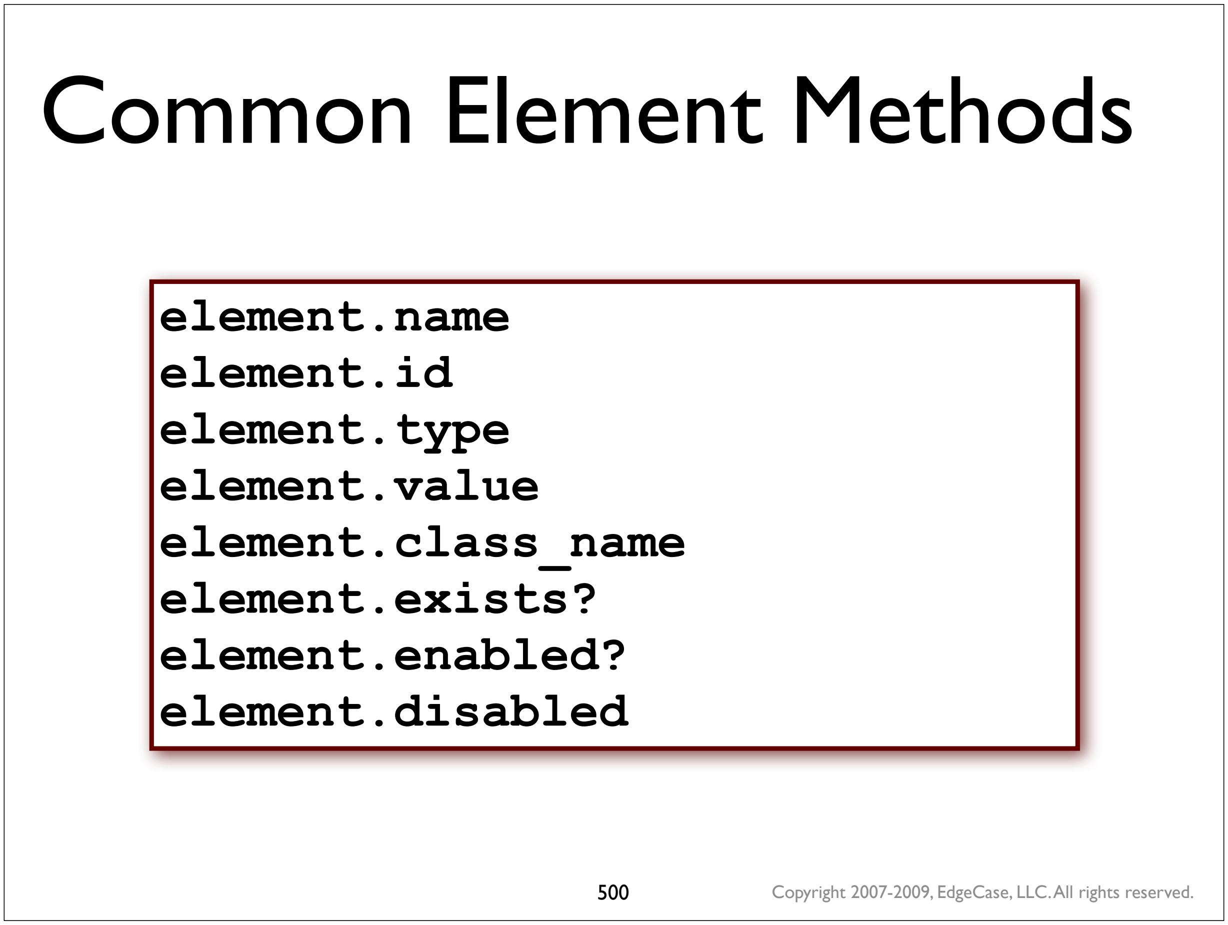


Finding Elements

```
ff.link(:index, 5)
ff.link(:text,  “click me”)
ff.link(:url,   “http://x.com”)
ff.link(:id,    “link_id”)
ff.link(:name,  “link_name”)
ff.link(:title, “link_title”)
ff.link(:xpath, “//a[...]”)
```

Regular Expressions may be used in place of strings.

Also, **:value** may be used with input fields.



Common Element Methods

```
element.name
element.id
element.type
element.value
element.class_name
element.exists?
element.enabled?
element.disabled
```

Element: link

```
link.click
link.href
```

Element: button

```
button.click
```

Element: checkbox/radio

```
checkbox.isSet?
checkbox.clear
checkbox.set
```

```
radio.isSet?
radio.clear
radio.set
```

Element: text_field

```
text_field.value
text_field.value=
```

Issues

- Sensitive to changes in the UI, tests are easily broken.
- Browser holds a lot of state, beware cross-test interference. (Particularly w.r.t. database)
- Sloooow ... runs at browsers speeds.

Selenium Testing

Selenium

- JavaScript-based
- In-browser testing
- Acts as a user of your application

```
script/plugin install
   http://svn.openqa.org/svn/selenium-on-rails/selenium-on-rails
```

perform sanity check

```
cd vendor/plugins

cd selenium-on-rails

rake
```

```
cd ../../../
(back to app root)
```

```
script/generate selenium make_reservation
```

Selenium parses the file into HTML using Redcloth (Textile)

The test runner only parses HTML tables as input so you can document away on each of your tests

```
|open|/|
|verifyTextPresent| Pragmatic Hotel||
|clickAndWait|new_link||
|verifyTextPresent|Room Availability||
```

Selenium parses the file into HTML using Redcloth (Textile)

The test runner only parses HTML tables as input so you can document away on each of your tests

```
|open|/|
|verifyTextPresent| Pragmatic Hotel||
|clickAndWait|new_link||
|verifyTextPresent|Room Availability||
```

Selenium parses the file into HTML using Redcloth (Textile)
The test runner only parses HTML tables as input so you can document away on each of your tests

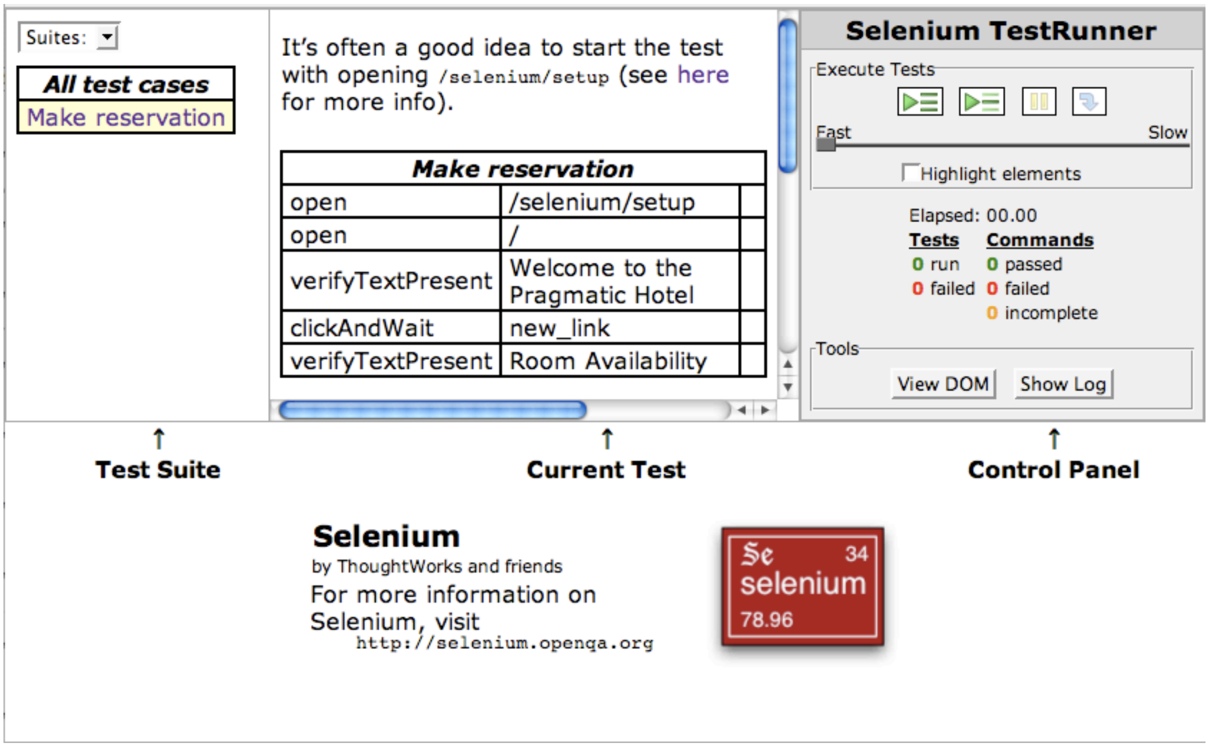
```
|open|/|
|verifyTextPresent| Pragmatic Hotel||
|clickAndWait|new_link||
|verifyTextPresent|Room Availability||
```

```
script/server -e test
```

```
copy config.yml.example
to config.yml
in vendor/plugins/selenium-on-rails
```

```
# rake test:acceptance settings
browsers:
  # Windows
  firefox: 'c:\Program Files\Mozilla Firefox\firefox.exe'
  ie: 'c:\Program Files\Internet Explorer\iexplore.exe'

  # Mac OS X
  #firefox: '/Applications/Firefox.app/Contents/MacOS/firefox-bin'
  #safari: '/Applications/Safari.app/Contents/MacOS/Safari'

#host: 'localhost'
#port_start: 3000
#port_end: 3005
#base_url_path: '/'
#max_browser_duration: 120
#multi_window: false
```

```
rake test:acceptance
```

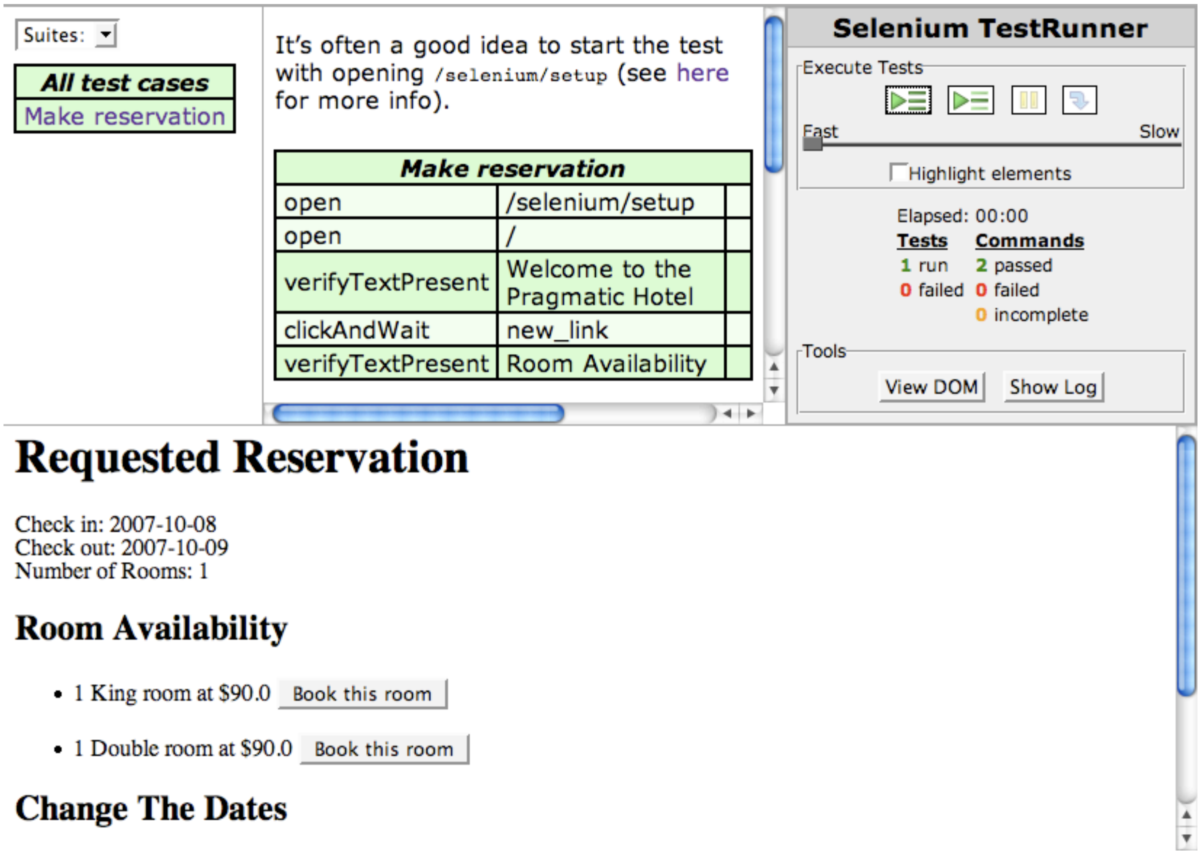


It's often a good idea to start the test with opening `/selenium/setup` (see here for more info).

Make reservation		
open	/selenium/setup	
open	/	
verifyTextPresent	Welcome to the Pragmatic Hotel	
clickAndWait	new_link	
verifyTextPresent	Room Availability	

Elapsed: 00.00

Tests		Commands	
0	run	0	passed
0	failed	0	failed
		0	incomplete



It's often a good idea to start the test with opening `/selenium/setup` (see here for more info).

Make reservation		
open	/selenium/setup	
open	/	
verifyTextPresent	Welcome to the Pragmatic Hotel	
clickAndWait	new_link	
verifyTextPresent	Room Availability	

Elapsed: 00:00

Tests	Commands
1 run	2 passed
0 failed	0 failed
	0 incomplete

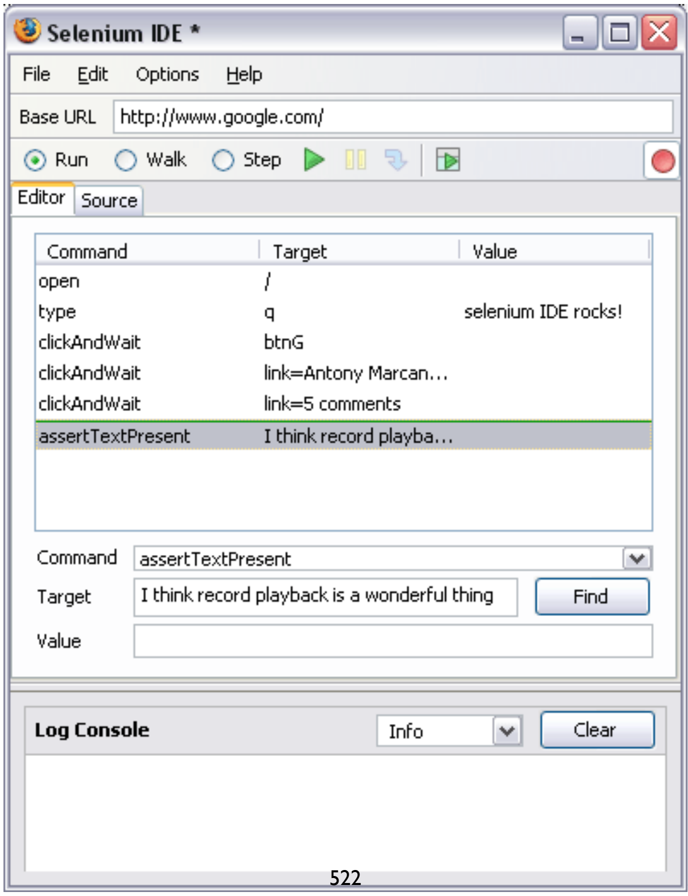
Requested Reservation

Check in: 2007-10-08
Check out: 2007-10-09
Number of Rooms: 1

Room Availability

- 1 King room at $90.0 Book this room
- 1 Double room at $90.0 Book this room

Change The Dates



Base URL: http://www.google.com/

Command	Target	Value
open	/	
type	q	selenium IDE rocks!
clickAndWait	btnG	
clickAndWait	link=Antony Marcan...	
clickAndWait	link=5 comments	
assertTextPresent	I think record playba...	

Command: assertTextPresent
Target: I think record playback is a wonderful thing
Value:

Selenese

"Language" for Selenium

- Actions
- Accessors
- Assertions
- Element Locators
- Variables

command	value	target
setVariable	base_url	'http://localhost:3000/'
setVariable	logout_url	'${base_url}/logout'
setVariable	signup_url	'${base_url}/signup'
open	${logout_url}	
open	${base_url}	
verifyTextPresent	Hello World	
click	//a[@href='${signup_url}']	
verifyTitle	Welcome - Please sign in	
verifyLocation	singup_form	
verifyTextPresent	Registration Form	

Selenese Options

- Selenese, .sel files
  - Standard Selenium commands
  - Table Based
  - HTML or Textile
- RSelenese, .rsel files
  - Ruby translation

webrat

demo

Cucumber

Behaviour Driven Development
with elegance and joy

External DSL
Integration Testing

Spec::UI

↓

Rspec Plain Text Stories

↓

Cucumber

```
[sudo] gem install cucumber
```

```
Feature: Test::Unit
  In order to please people who like Test::Unit
  As a Cucumber user
  I want to be able to use assert* in my
  step definitions

  Scenario: assert_equal
    Given x = 5
    And y = 5
    Then I can assert that x == y
```

```
Feature: Test::Unit
  In order to please people who like Test::Unit
  As a Cucumber user
  I want to be able to use assert* in my
  step definitions
```

The User Story

```
  Scenario: assert_equal
    Given x = 5
    And y = 5
    Then I can assert that x == y
```

```
Feature: Test::Unit
  In order to please people who like Test::Unit
  As a Cucumber user
  I want to be able to use assert* in my
  step definitions
```

```
  Scenario: assert_equal
    Given x = 5
    And y = 5
    Then I can assert that x == y
```

The Scenario

Feature: Test::Unit
  In order to please people who like Test::Unit
  As a Cucumber user
  I want to be able to use assert* in my
  step definitions

The User Story

- Feature definitions are at the beginning of the file.
- Can contain any text.
- Ends with word ‘Scenario’ on separate line

great opportunity for documentation

```
Scenario: assert_equal
  Given x = 5
  And y = 5
  Then I can assert that x == y
```

The Scenario

- Scenario's begin with word **Scenario**
- Words after it are used as a description
- Followed by a list of steps

Steps

Given

When

Then

But

And

Steps

Scenario: assert_equal

Given x = 5

And y = 5

Then I can assert that x == y

Steps

Scenario: assert_equal

Given x = 5

And y = 5

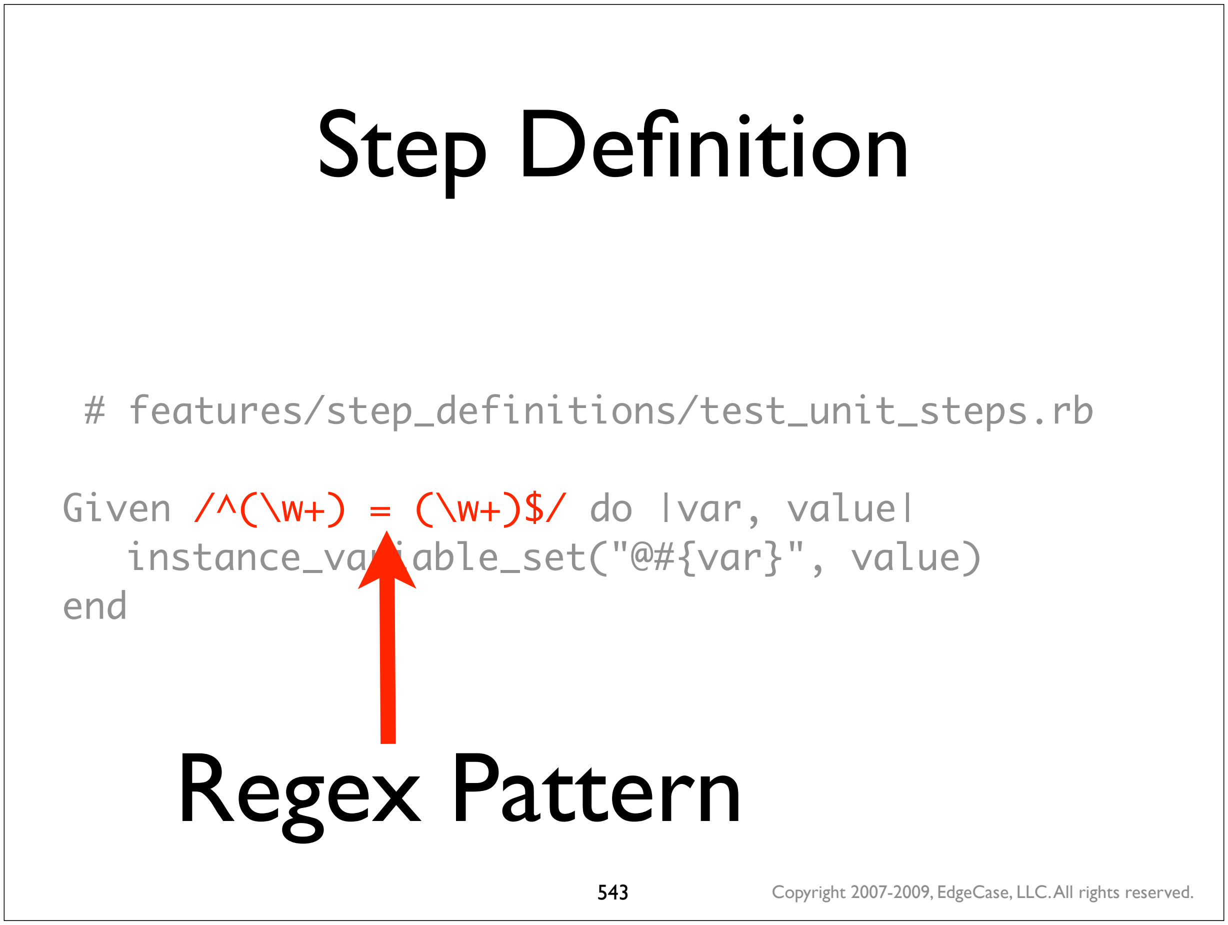
Then I can assert that x == y

Steps

Scenario: assert_equal
Given x = 5
And y = 5
Then I can assert that x == y

Step Definition

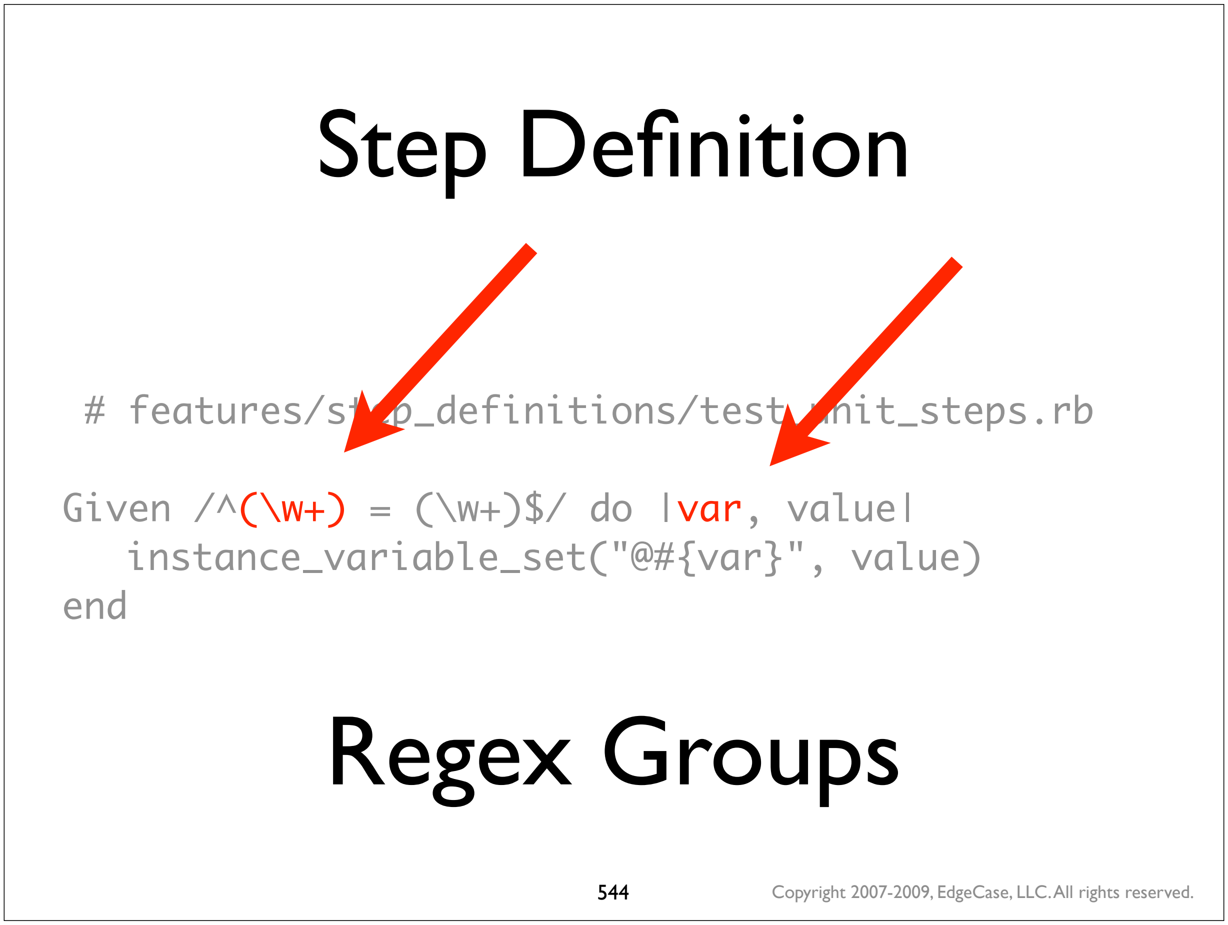
```
 # features/step_definitions/test_unit_steps.rb

Given /^(\w+) = (\w+)$/ do |var, value|
   instance_variable_set("@#{var}", value)
end
```



Step Definition

```
# features/step_definitions/test_unit_steps.rb

Given /^(\w+) = (\w+)$/ do |var, value|
  instance_variable_set("@#{var}", value)
end
```

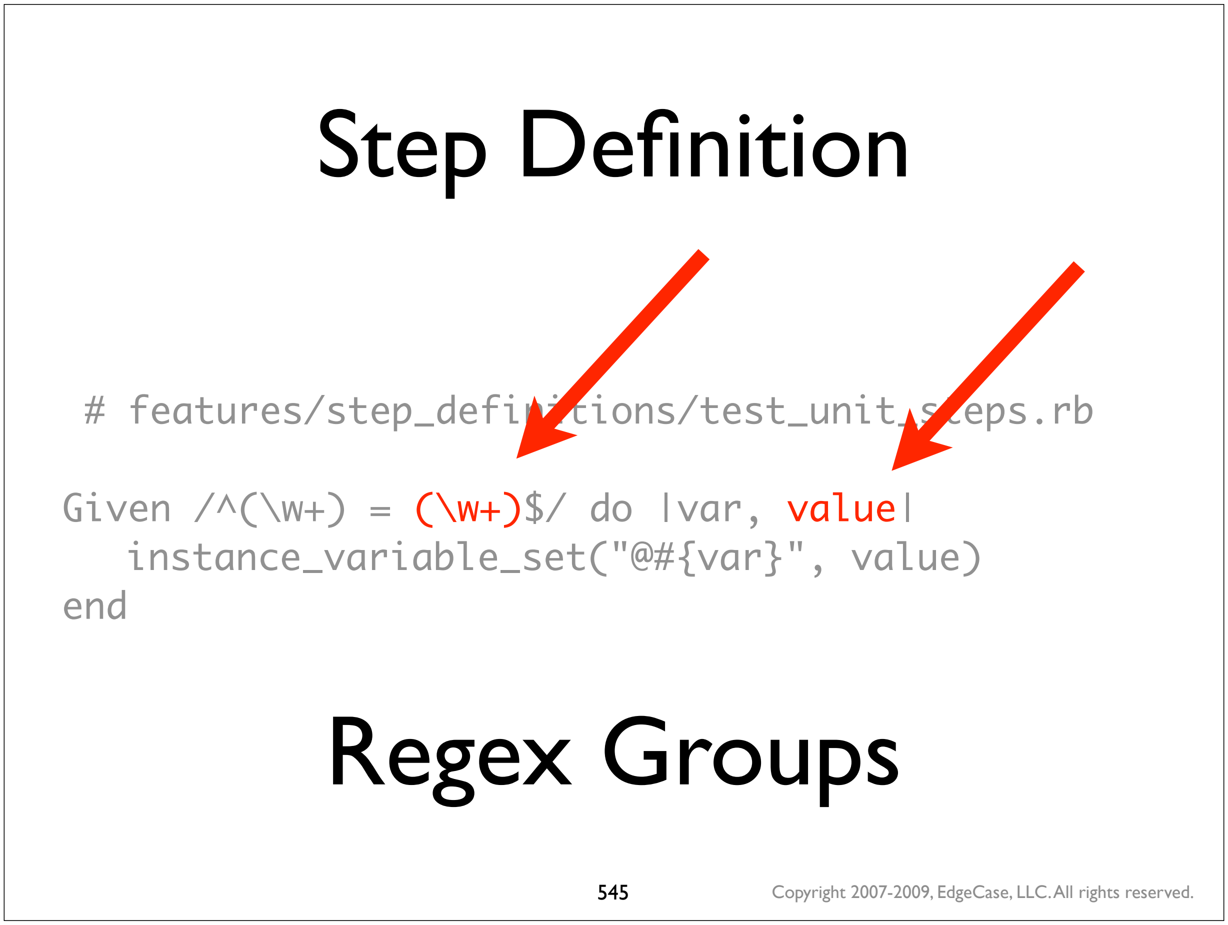
Regex Pattern



Step Definition

```
# features/step_definitions/test_unit_steps.rb

Given /^(\w+) = (\w+)$/ do |var, value|
  instance_variable_set("@#{var}", value)
end
```

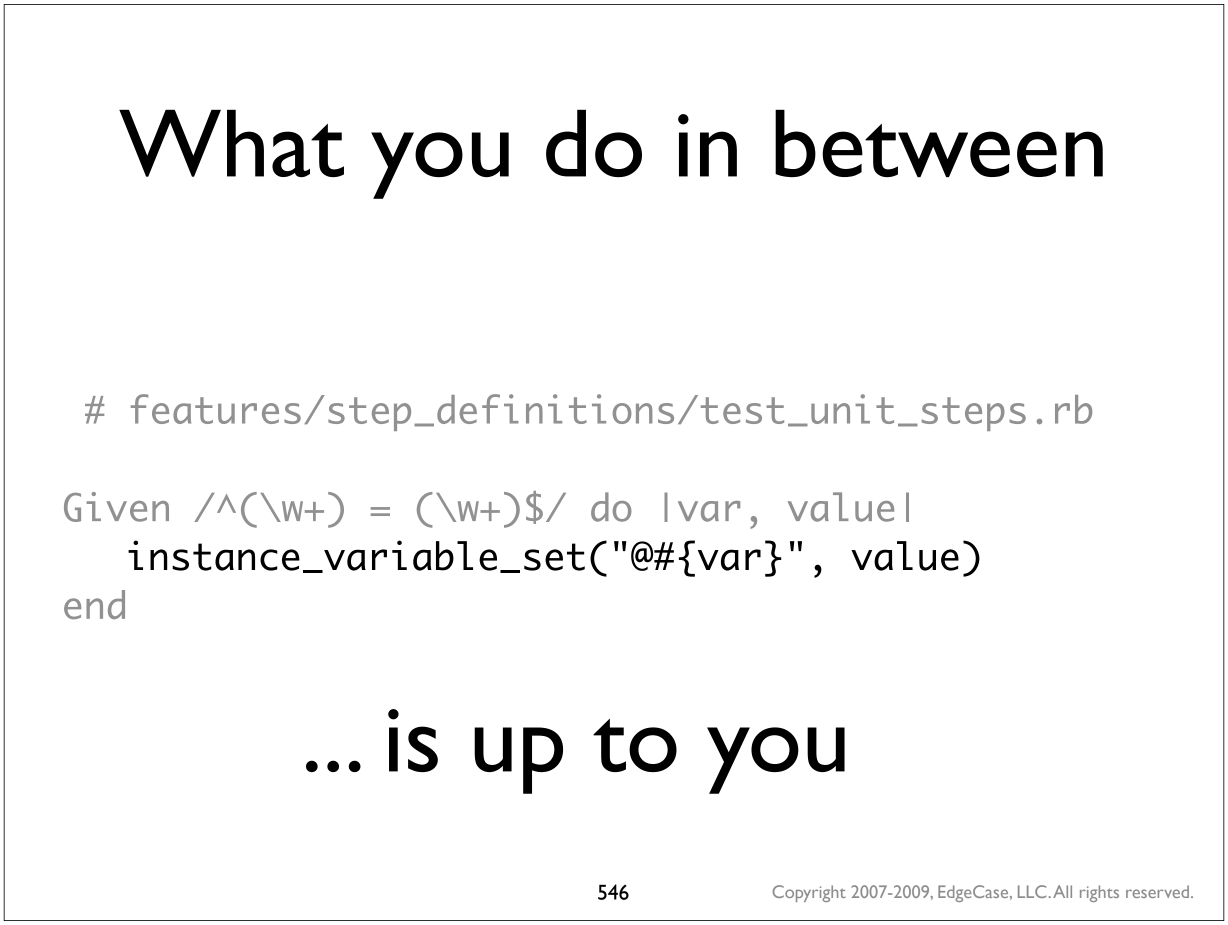
Regex Groups



Step Definition

```
# features/step_definitions/test_unit_steps.rb

Given /^(\w+) = (\w+)$/ do |var, value|
  instance_variable_set("@#{var}", value)
end
```

Regex Groups



What you do in between

```
# features/step_definitions/test_unit_steps.rb

Given /^(\w+) = (\w+)$/ do |var, value|
  instance_variable_set("@#{var}", value)
end
```

... is up to you

It’s a whole new world

```
World do
  @object = Object.new
end
```

... is up to you

Hooks

```
Before do
  @calc = Calculator.new
end

After do |scenario|
  if scenario.status.index(:failed)
    # Call the BDD police
  end
end
```

Scenarios

```
Scenario Outline: Add two numbers
  Given I have entered <input_1> into the calculator
  And I have entered <input_2> into the calculator
  When I press <button>
  Then the result should be <output> on the screen

Examples:
  | input_1 | input_2 | button | output |
  | 20      | 30      | add    | 50     |
  | 2       | 5       | add    | 7      |
  | 0       | 40      | add    | 40     |
```

Scenarios

```
Scenario Outline: Add two numbers
  Given I have entered <input_1> into the calculator
  And I have entered <input_2> into the calculator
  When I press <button>
  Then the result should be <output> on the screen

Examples:
  | input_1 | input_2 | button | output |
  | 20      | 30      | add    | 50     |
  | 2       | 5       | add    | 7      |
  | 0       | 40      | add    | 40     |
```

Scenarios

```
Scenario Outline: Add two numbers
  Given I have entered <input_1> into the calculator
  And I have entered <input_2> into the calculator
  When I press <button>
  Then the result should be <output> on the screen

Examples:
  | input_1 | input_2 | button | output |
  | 20      | 30      | add    | 50     |
  | 2       | 5       | add    | 7      |
  | 0       | 40      | add    | 40     |
```

Scenarios

```
Scenario Outline: Add two numbers
  Given I have entered <input_1> into the calculator
  And I have entered <input_2> into the calculator
  When I press <button>
  Then the result should be <output> on the screen

Examples:
  | input_1 | input_2 | button | output |
  | 20      | 30      | add    | 50     |
  | 2       | 5       | add    | 7      |
  | 0       | 40      | add    | 40     |
```

Scenarios

```
Scenario Outline: Add two numbers
  Given I have entered <input_1> into the calculator
  And I have entered <input_2> into the calculator
  When I press <button>
  Then the result should be <output> on the screen

Examples:
  | input_1 | input_2 | button | output |
  | 20      | 30      | add    | 50     |
  | 2       | 5       | add    | 7      |
  | 0       | 40      | add    | 40     |
```

rspec, test/unit, selenium or watir?

yes!

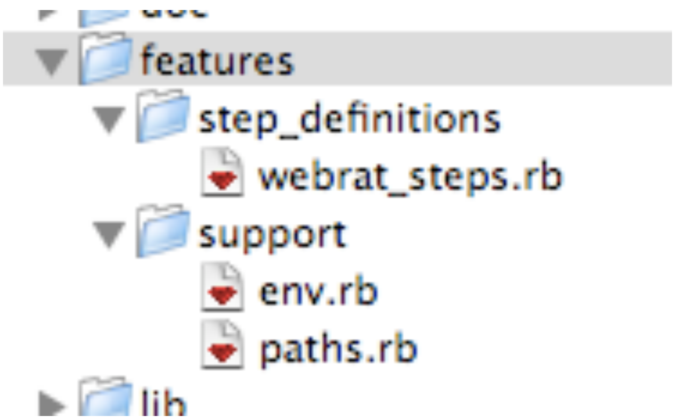
```
Given 'I am on the Google search page' do
  @browser.goto 'http://www.google.com/'
end

When /I search for "(.*)"/ do |query|
  @browser.text_field(:name, 'q').set(query)
  @browser.button(:name, 'btnG').click
end

Then /I should see a link to "(.*)":(.*)/ do |text, url|
  link = @browser.link(:url, url)
  link.should_not == nil
  link.text.should == text
end
```

Cucumber

Behaviour Driven Development
with elegance and joy

on Rails

```
script/generate cucumber
```



```
doc
features
  step_definitions
    webrat_steps.rb
  support
    env.rb
    paths.rb
lib
```

```
script/generate feature
```

`script/generate feature`

“Don’t get addicted to this generator – you’re better off writing these by hand in the long run.”

- github documentation

now you choose ...

- watir
- webrat
- selenium

JavaScript Testing

Legacy Testing

metric-fu

Test Driven Development with Rails

Lab 9: Integration Tests

Lab 9: Integration Tests

Pick a method and create integration tests for app

`Rails Integration | Watir | Selenium`

- ✓ How much functionality can you test?
- ✓ How readable can you make your test?

Lab 9: Integration Tests

Pick a method and create integration tests for app

`Rails Integration | Watir | Selenium`

- ✓ How much functionality can you test?
- ✓ How readable can you make your test?

- ✓ Pick another method and repeat

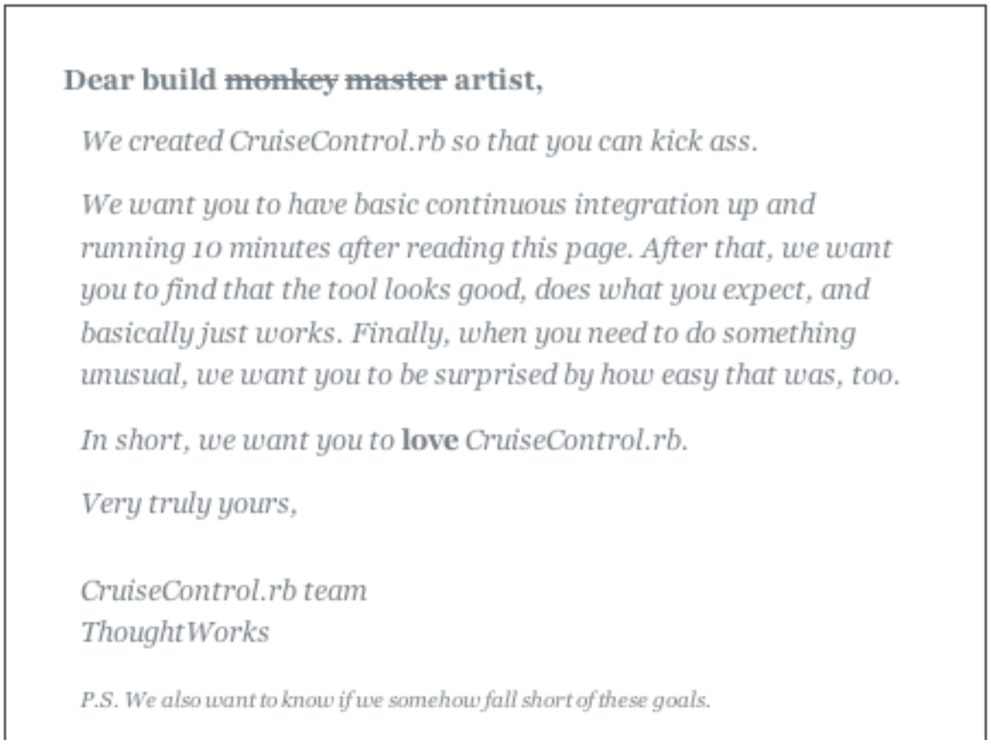
Lab 8: Spec the application

- ✓ Examples in the application (spec directory)
- ✓ See how much you can spec out
- ✓ Spec at least one of each:
  - ✓ Controller
  - ✓ View
  - ✓ Model
  - ✓ spec/ui (integration)

Test Driven Development with Rails

Continuous Integration

CruiseControl.rb



Dear build ~~monkey~~ ~~master~~ artist,

We created CruiseControl.rb so that you can kick ass.

We want you to have basic continuous integration up and running 10 minutes after reading this page. After that, we want you to find that the tool looks good, does what you expect, and basically just works. Finally, when you need to do something unusual, we want you to be surprised by how easy that was, too.

In short, we want you to **love** *CruiseControl.rb.*

Very truly yours,

CruiseControl.rb team
ThoughtWorks

P.S. We also want to know if we somehow fall short of these goals.

http://cruisecontrolrb.thoughtworks.com/documentation/getting_started

DEMO

Setup

- Grab lastest from GitHub
  - http://github.com/thoughtworks/cruisecontrol.rb
- cd into the cruise directory
- Add the project
  - ./cruise add *projname* -r *projurl*
- Configure/create/migrate the DB
- Make sure it builds
- Run: cruise start
- Goto http://localhost:3333 and click 'build'

How does it know what to build?

- Looks for a rake file and one of the following sets of tasks:
  - cruise
  - db:test:purge, db:migrage, test
  - default
- Custom rake targets or non-rake commands can be used for the buile

Configuration

- Edit `cruise/projects/`*`projname`*`/cruise_config.rb`
- Configure
  - Emails to get notification
  - Custom rake targets or build commands to use to build the target.
  - Polling interval
- More configuration in `cruise/config` directory

Monitoring

- Easy monitoring via EMail
- RSS feeds available from web server
- Works with CCTray (windows only)

Limitations

- The cruise Builder and WebServer must run on the same box.
  - However, it does support remote builds via ssh
- Only supports subversion for version control.

Notice that "ruby-only projects" is ***not*** *one of the limitations, so feel free to use CruiseControl.rb on your Java or .NET projects.*

Other CI Options

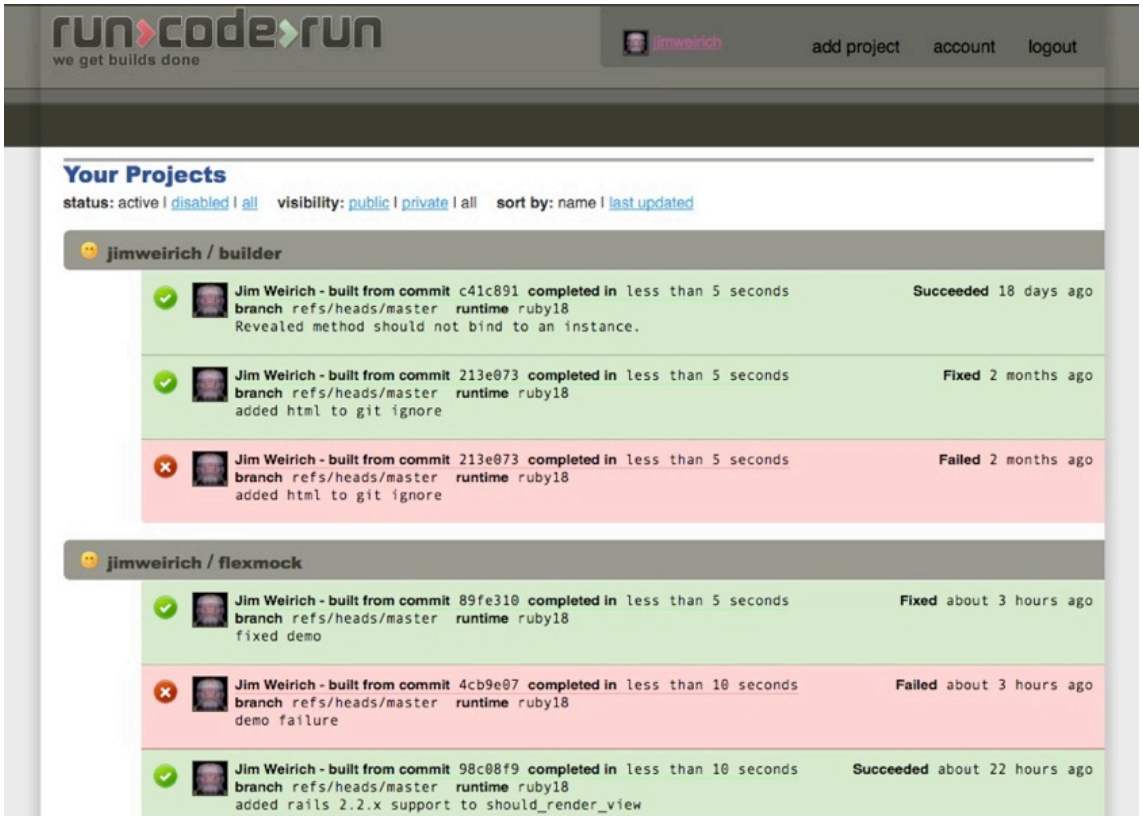
- Integrity
  - Minimal set of features
  - git only
  - Ruby based
  - http://integrityapp.com/

Other CI Options

- Hudson
  - Simple to setup
  - complete set of features
  - Java based
  - https://hudson.dev.java.net/

Other CI Options

- Run Code Run (https://runcoderun/)
  - Simple to setup
  - GitHub Only???
  - Hosted by 3rd Party
    - Free to Open Source
    - Beta for private repos



Your Projects

status: active | disabled | all **visibility:** public | private | all **sort by:** name | last updated

jimweirich / builder

```
✓ Jim Weirich - built from commit c41c891 completed in less than 5 seconds      Succeeded 18 days ago
  branch refs/heads/master  runtime ruby18
  Revealed method should not bind to an instance.

✓ Jim Weirich - built from commit 213e073 completed in less than 5 seconds      Fixed 2 months ago
  branch refs/heads/master  runtime ruby18
  added html to git ignore

✗ Jim Weirich - built from commit 213e073 completed in less than 5 seconds      Failed 2 months ago
  branch refs/heads/master  runtime ruby18
  added html to git ignore
```

jimweirich / flexmock

```
✓ Jim Weirich - built from commit 89fe310 completed in less than 5 seconds      Fixed about 3 hours ago
  branch refs/heads/master  runtime ruby18
  fixed demo

✗ Jim Weirich - built from commit 4cb9e07 completed in less than 10 seconds     Failed about 3 hours ago
  branch refs/heads/master  runtime ruby18
  demo failure

✓ Jim Weirich - built from commit 98c08f9 completed in less than 10 seconds     Succeeded about 22 hours ago
  branch refs/heads/master  runtime ruby18
  added rails 2.2.x support to should_render_view
```

Resources

- Alumni Mailing List
- Job Board
- Yearbook

Questions?

The Pragmatic Studio

Thank you!