fixture_replacement

aka I’d rather eat fire ants than maintain relational test data in yaml

Fixtures Suck.

No validation

Inconsistent load order

Brittle

Test specific

Relationship maintenance **nightmare**

Incomplete tear-down

Slow

The Overnight Scenario

FixtureScenario

Tom Preston-Werner (aka mojombo) (aka god)

Contextualize by putting fixtures into subdirectories

Each subdirectory represents a context

i.e. users/*.yml, users/banned/*.yml

```
scenario :banned
```

Fixin’ Fixtures

FixtureScenarioBuilder

aka the Factory Factory

Chris Wanstrath o’ Err The Blog

Pour some Ruby on it

```ruby
scenario :blog do
  %w( Tom Chris Kevin ).each_with_index do |user, index|
    User.create! :name => user, :banned => index.odd?
  end
end
```

Refactory

```ruby
module Factory
  def self.create_customer(attributes = {})
    default_attributes = {
      :first_name => "Dan",
      :last_name  => "Manges",
      :paperboy   => create_paperboy
    }
    Customer.create! default_attributes.merge(attributes)
  end

  def self.create_newspaper(attributes = {})
    default_attributes = {
      :customer => create_customer,
      :headline => "Read all about it!"
      :paperboy => create_paperboy
    }
    Newspaper.create! default_attributes.merge(attributes)
  end

  def self.create_paperboy(attributes = {})
    default_attributes = {
      :first_name     => "Paper",
      :last_name      => "Boy",
      :delivery_route => "Main St Route"
    }
    Paperboy.create! default_attributes.merge(attributes)
  end
end
```

Object Factory

by Dan Mange$

Provides defaults

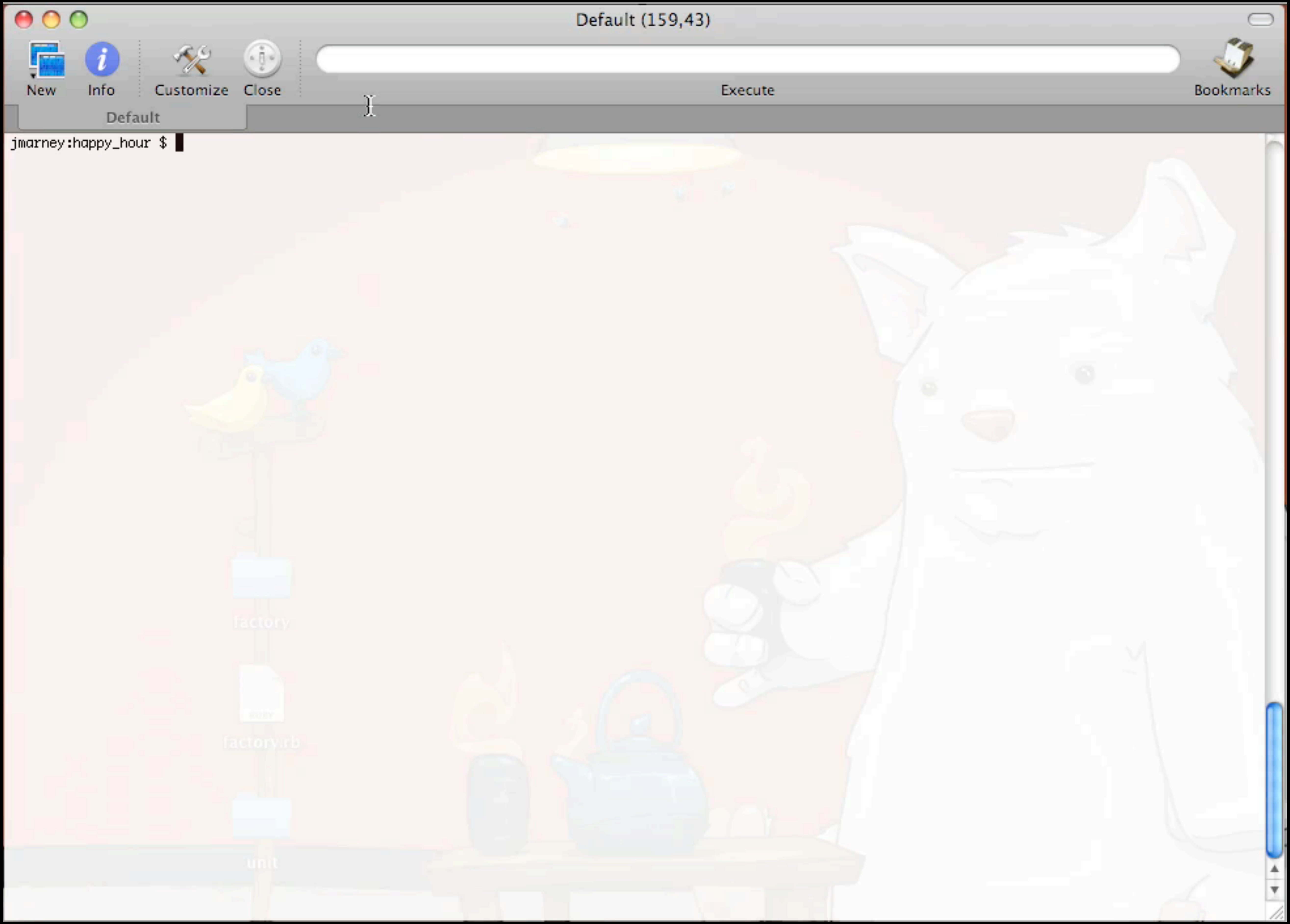
Always valid

Improved readability

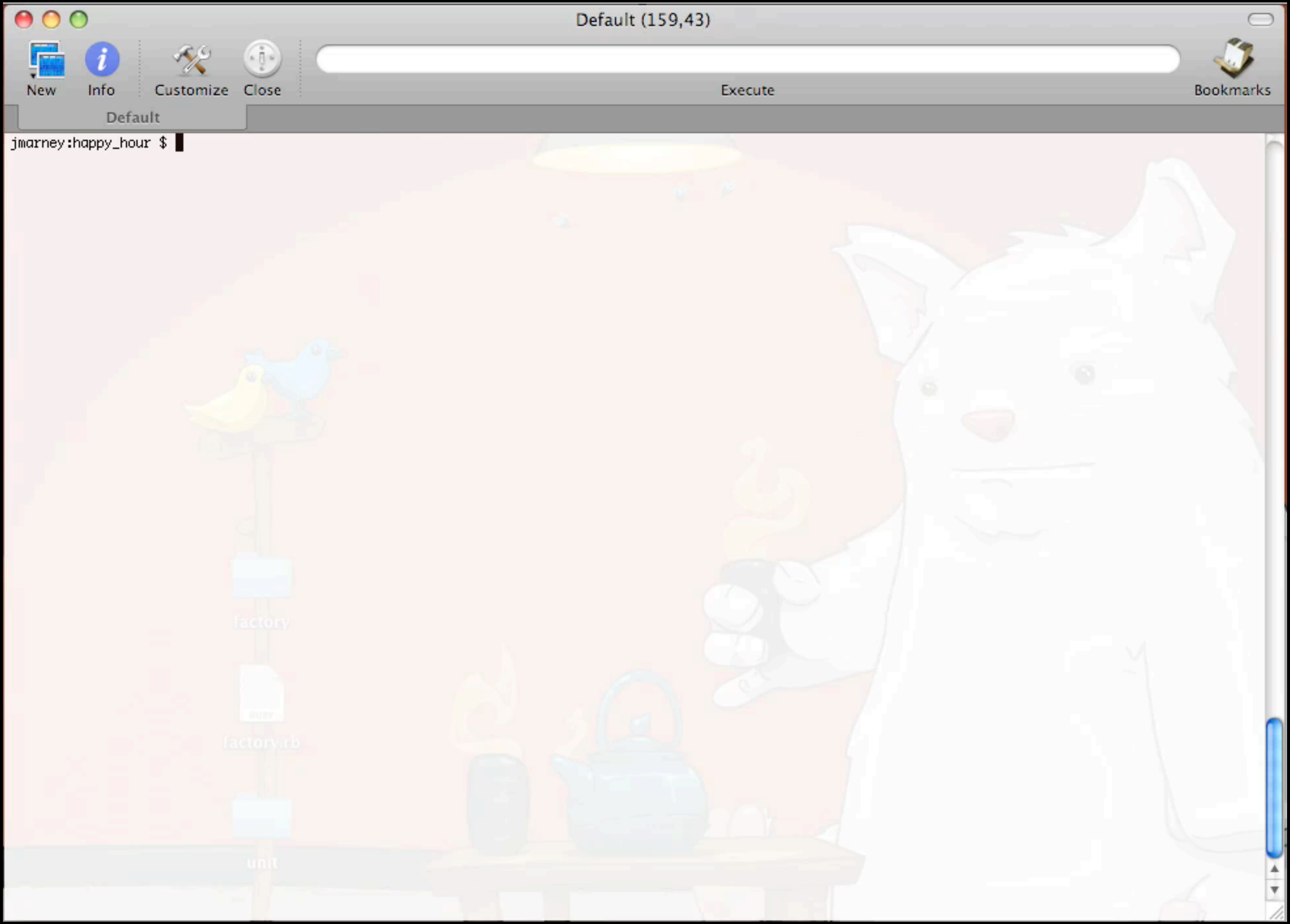
Consistent state

It’s not you, it’s me

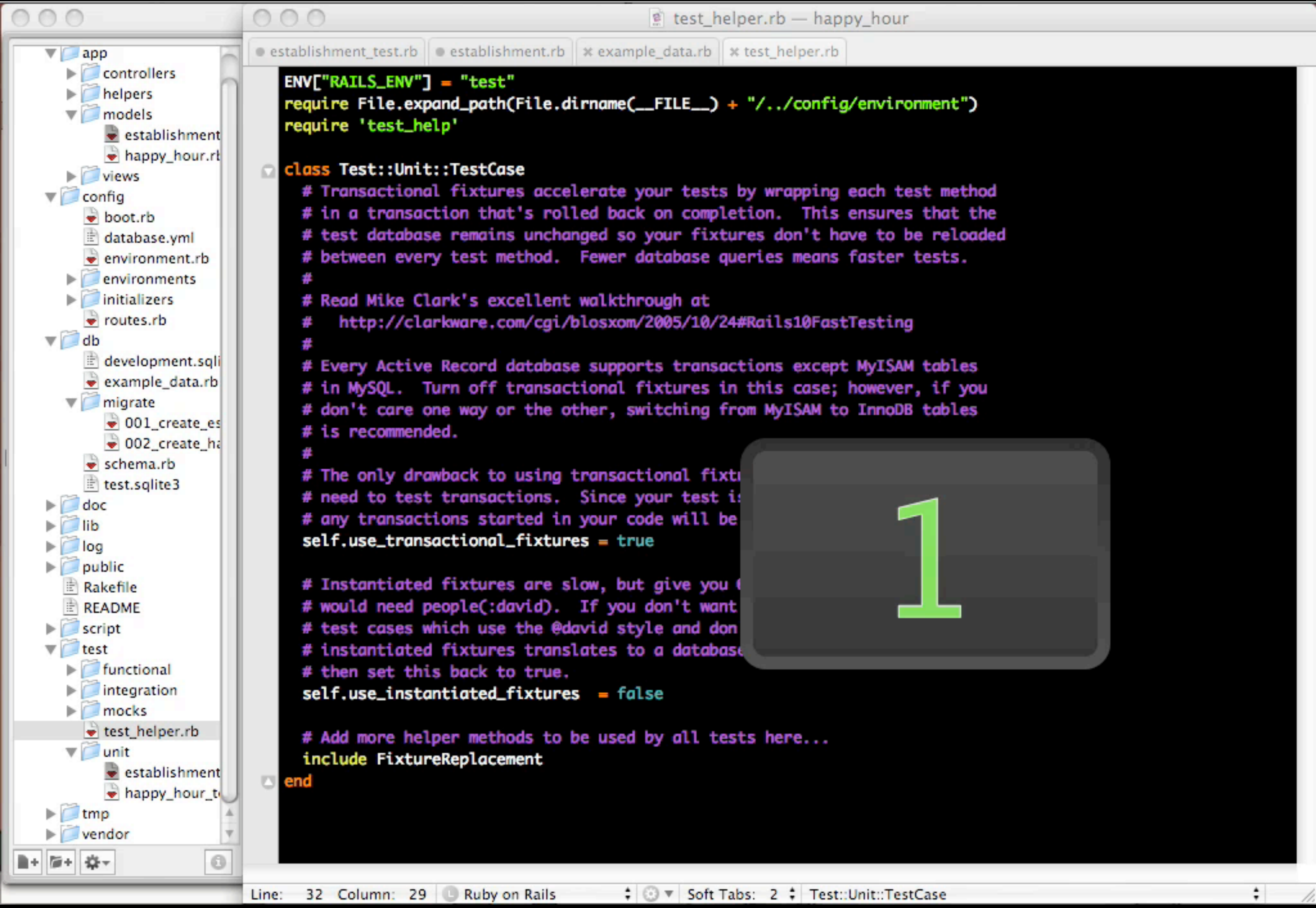
aka breaking up with fixtures



```
jmarney:happy_hour $ 
```



```
jmarney:happy_hour $
```

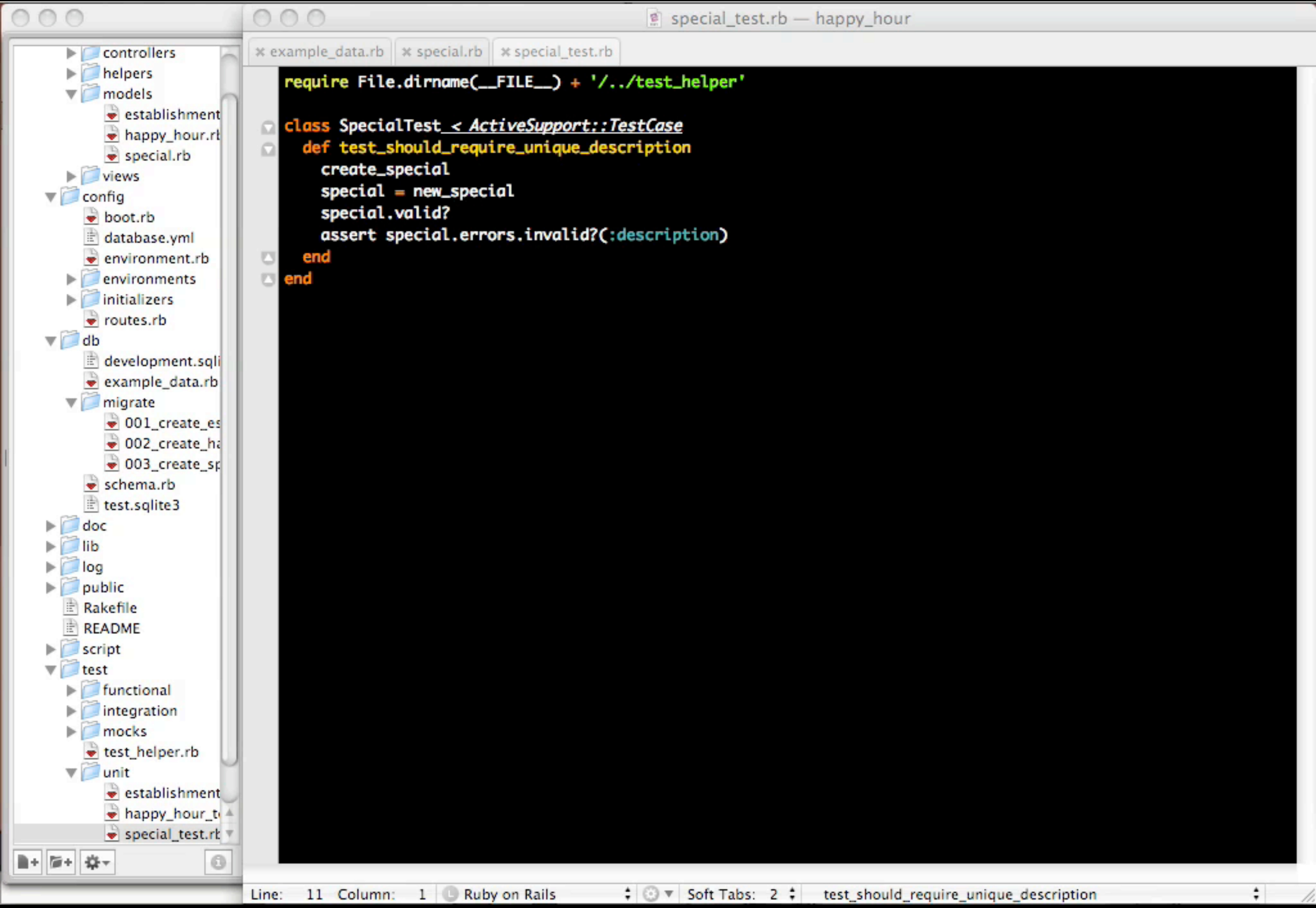


```ruby
ENV["RAILS_ENV"] = "test"
require File.expand_path(File.dirname(__FILE__) + "/../config/environment")
require 'test_help'

class Test::Unit::TestCase
  # Transactional fixtures accelerate your tests by wrapping each test method
  # in a transaction that's rolled back on completion.  This ensures that the
  # test database remains unchanged so your fixtures don't have to be reloaded
  # between every test method.  Fewer database queries means faster tests.
  #
  # Read Mike Clark's excellent walkthrough at
  #   http://clarkware.com/cgi/blosxom/2005/10/24#Rails10FastTesting
  #
  # Every Active Record database supports transactions except MyISAM tables
  # in MySQL.  Turn off transactional fixtures in this case; however, if you
  # don't care one way or the other, switching from MyISAM to InnoDB tables
  # is recommended.
  #
  # The only drawback to using transactional fixt
  # need to test transactions.  Since your test i
  # any transactions started in your code will be
  self.use_transactional_fixtures = true

  # Instantiated fixtures are slow, but give you
  # would need people(:david).  If you don't want
  # test cases which use the @david style and don
  # instantiated fixtures translates to a databas
  # then set this back to true.
  self.use_instantiated_fixtures  = false

  # Add more helper methods to be used by all tests here...
  include FixtureReplacement
end
```



special_test.rb — happy_hour

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

class SpecialTest < ActiveSupport::TestCase
  def test_should_require_unique_description
    create_special
    special = new_special
    special.valid?
    assert special.errors.invalid?(:description)
  end
end
```

Line: 11 Column: 1 Ruby on Rails Soft Tabs: 2 test_should_require_unique_description

Learnmedy

Tom Preston-Werner (aka mojombo)
http://code.google.com/p/fixture-scenarios/
http://rubyisawesome.com/
http://god.rubyforge.org/

Chris Wanstrath
http://errtheblog.com/posts/61-fixin-fixtures
http://ozmm.org/

Dan Manges
http://www.dcmanges.com/blog/38

fixture_replacement
http://replacefixtures.rubyforge.org/