Rails Testing

Sang Shin
Michèle Garoche
www.javapassion.com
"Learning is fun!"

Topics

- Overview of Rails testing
- Unit testing
- Functional testing
- Integration testing
- RSpec

Overview of Rails Testing

Types of Testing in Rails

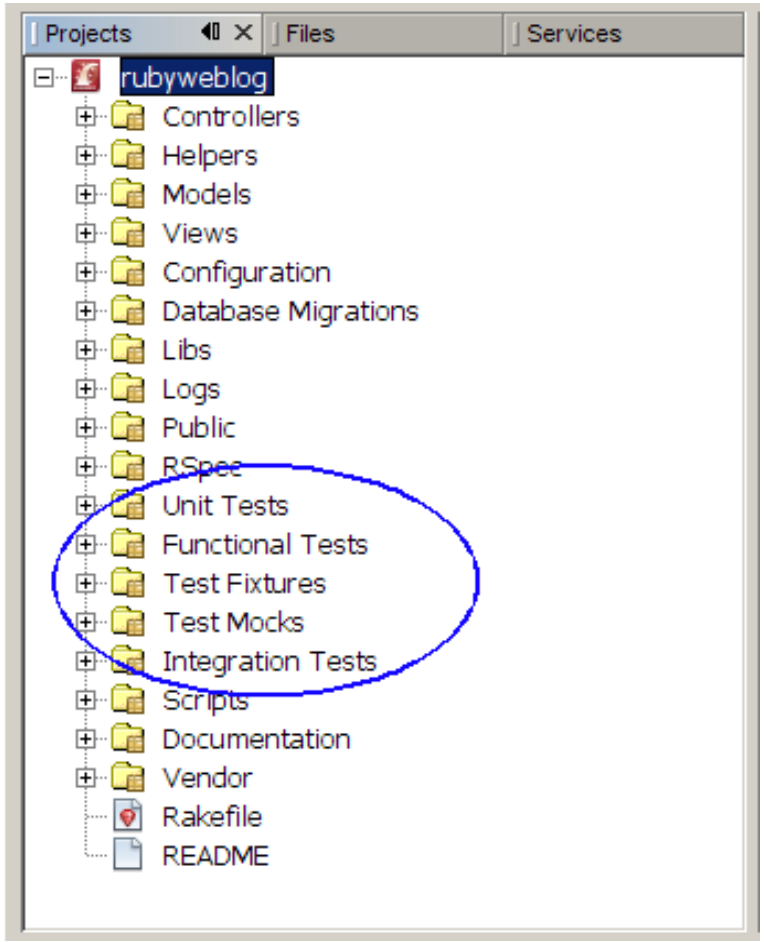
- Unit testing
  - Tests models
- Functional testing
  - Tests controllers
- Integration testing
  - Tests multiple controllers

What Does Rails Provide for Testing?

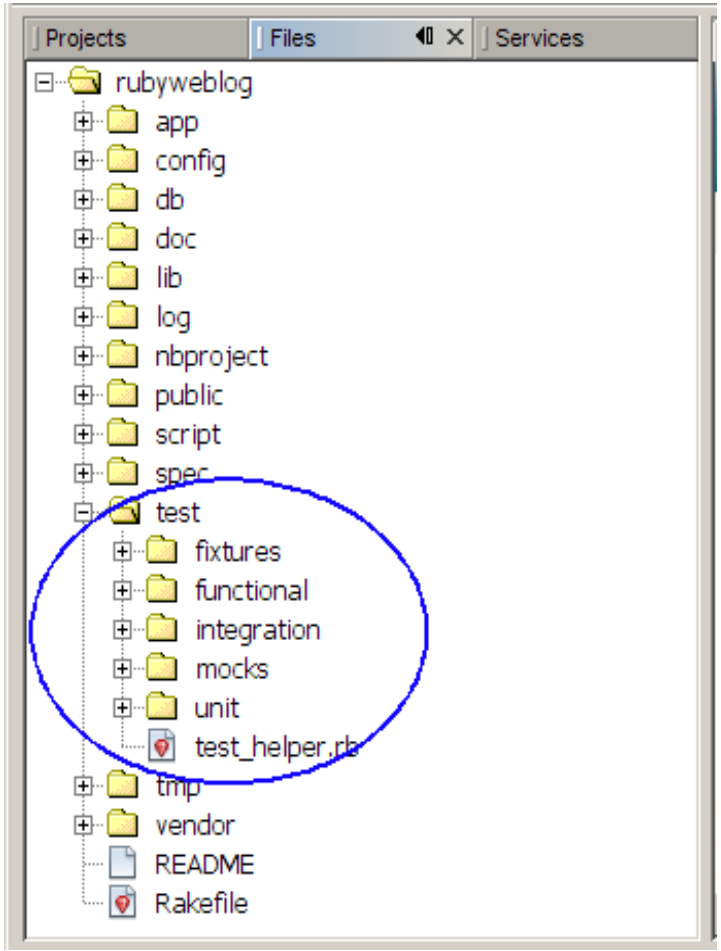
- Create Test directories
  - Unit Tests (test/unit)
  - Functional Tests (test/functional)
  - Integration Tests (test/integration)
  - Test Mocks (test/mocks)
  - Test Fixtures (test/fixtures)
- An environment for testing
  - You can specify test database in the database.yml file

Test Directories Created by Rails

Logical directories



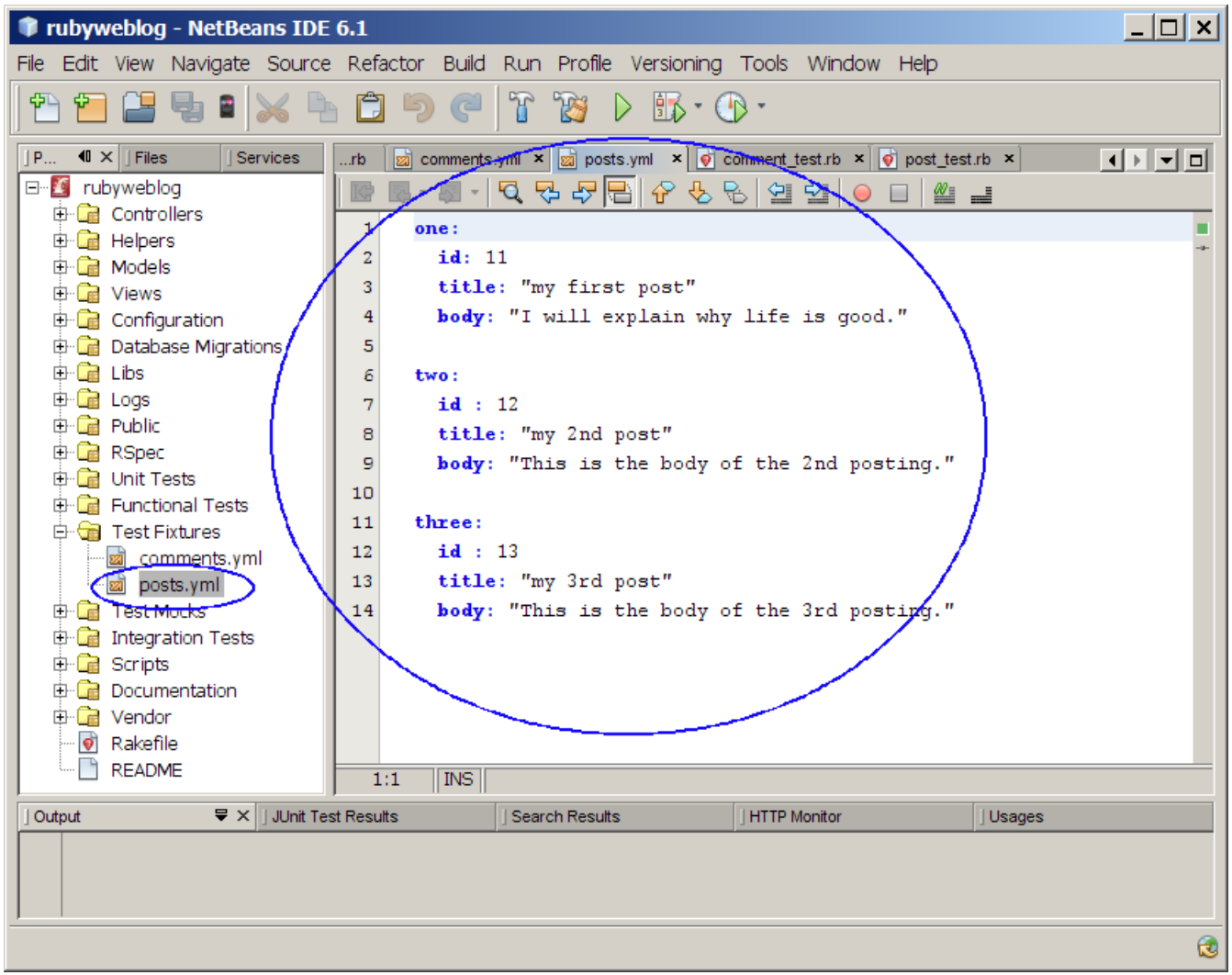
Physical



Fixtures

- Textual representations of table data
- Is used to populate the database before testing

Fixtures: Example



Mocks

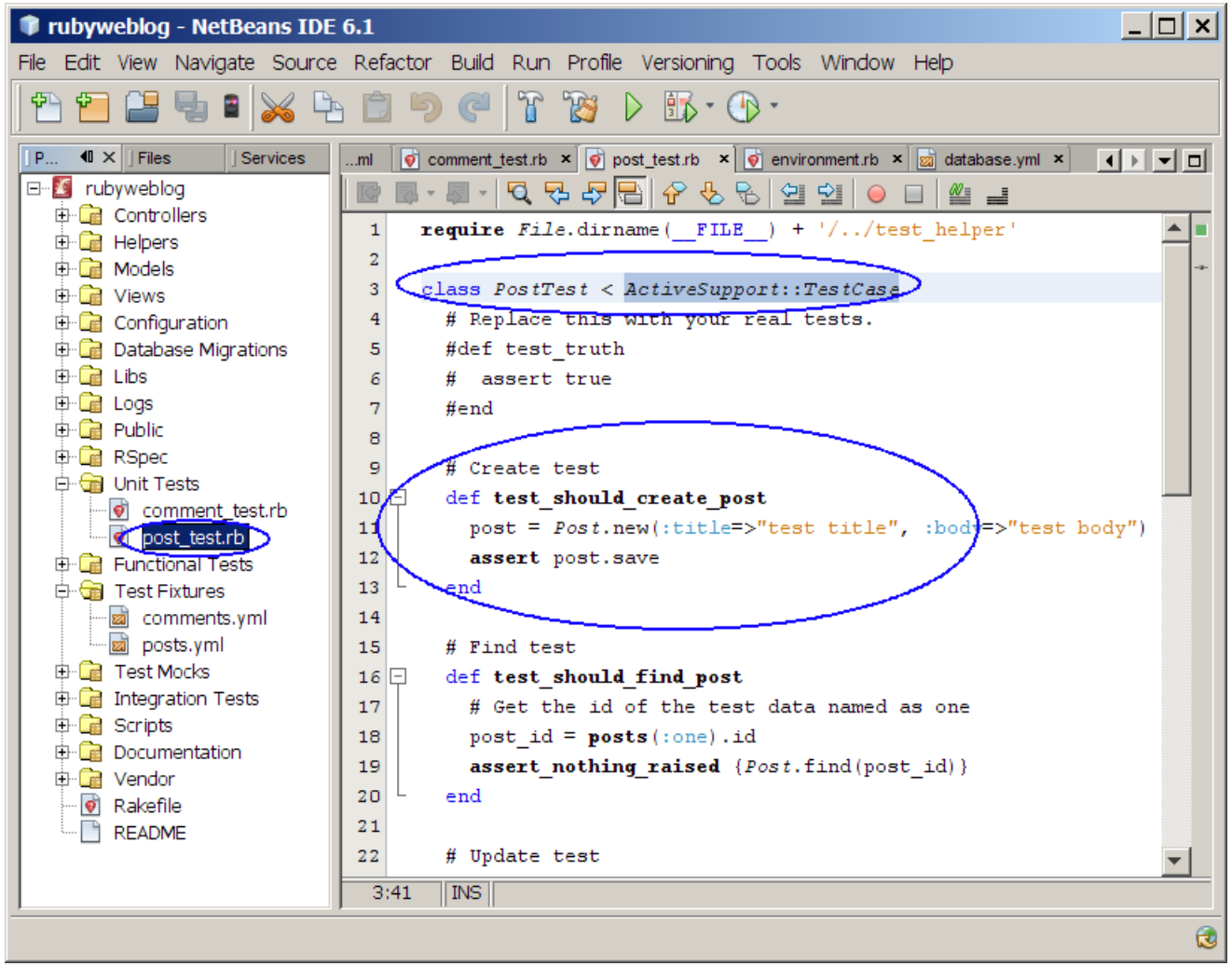
- Classes that are used in place of the actual classes while testing is being performed
  - Useful when using a real object is difficult, time-consuming, or hard to generate.
  - A mock object is used to test the behavior of another object since it mimics the behavior of the real object in a controlled manner.
  - A good example of a mock object is a dummy used instead of a real person in car crash testing.

Unit Testing: Testing a Model

Unit Testing Class

- Subclass of *ActiveSupport::TestCase* class
  - *class PostTest < ActiveSupport::TestCase*
- It requires test_helper
  - *require File.dirname(__FILE__)* + '*/../test_helper*'
- Test methods are prefixed with *test_*
  - *test_my_method*
- Within test methods, assert methods are used to test the result
  - assert methods asserts that the result is true

Unit Testing Class: Example



Unit Testing Examples

```
# Create test
def test_should_create_post
  post = Post.new(:title=>"test title", :body=>"test body")
  # make sure save is successful
  assert post.save
end

# Find test
def test_should_find_post
  # Get the id of the test data named as one
  post_id = posts(:one).id
  # make sure find is successful by checking an exception
  assert_nothing_raised {Post.find(post_id)}
end
```

Unit Testing Examples

```
# Destroy test
def test_should_destroy_post
  post = posts(:three)
  post.destroy
  # make sure find results RecordNotFound error
  assert_raise(ActiveRecord::RecordNotFound) {
                                  Post.find(post.id)
  }
end
```

Unit Testing Examples

```
# Validation test
def test_should_not_create_invalid_post
  post = Post.new()
  assert !post.valid?
  assert_equal "can't be blank", post.errors.on(:title)
  assert !post.save
end

# Association test
def test_should_find_two_comments
  post = posts(:one)
  assert_equal post.comments.find(:all).size, 2
  assert_equal post.comment_ids, [31, 32]
  assert_equal post.comments.find(:first).id, 31
end
```

Functional Testing: Testing a Controller

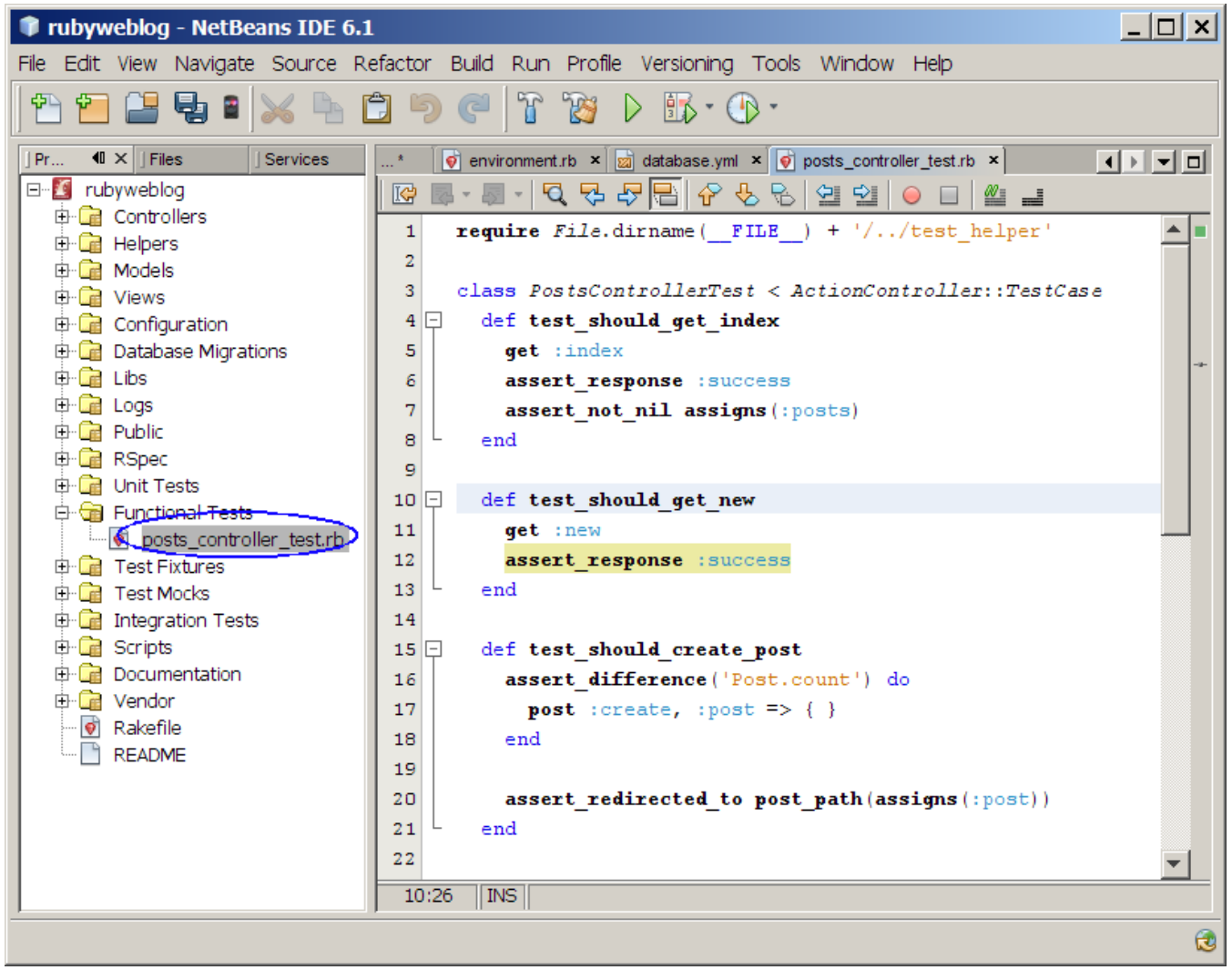
Functional Testing in Rails

- It lets you test your application in the context of Web application
  - Functional testing tests request/response of a controller
- Rails sets up the request and response objects for the testing
  - They act like real request and response

Functional Test Class

- Subclass of *ActionController::TestCase* class
  - *class PostsControllerTest < ActionController::TestCase*
- It requires test_helper
  - *require File.dirname(__FILE__) + '/../test_helper'*

Functional Test Class



Functional Testing Examples

```
def test_should_get_index
  get :index
  assert_response :success
  assert_not_nil assigns(:posts)
end

def test_should_get_new
  get :new
  assert_response :success
end
```

Functional Testing Examples

```
def test_should_create_post
  assert_difference('Post.count') do
    post :create, :post => { }
  end

  assert_redirected_to post_path(assigns(:post))
end

def test_should_show_post
  get :show, :id => posts(:one).id
  assert_response :success
end
```

Functional Testing Examples

```
def test_should_get_index
  get :index
  assert_response :success
  assert_not_nil assigns(:posts)
end

def test_should_get_new
  get :new
  assert_response :success
end

def test_should_get_edit
  get :edit, :id => posts(:one).id
  assert_response :success
end
```

Functional Testing Examples

```
def test_should_update_post
  put :update, :id => posts(:one).id, :post => { }
  assert_redirected_to post_path(assigns(:post))
end

def test_should_destroy_post
  assert_difference('Post.count', -1) do
    delete :destroy, :id => posts(:one).id
  end

  assert_redirected_to posts_path
end
```

Integration Testing: Testing Multiple Controllers

Integration Testing

- Spans multiple controllers with session support

Integration Test Class

- Subclass of *ActionController::IntegrationTest* class
  - *class CreatingNewPostAndCommentTest < ActionController::IntegrationTest*
- It requires test_helper
  - *require "#{File.dirname(__FILE__)}/../test_helper*"

Integration Testing Example

```
def test_should_create_1_post_and_2_comments
  post '/posts/create', :post => {:title=>"Rails", :body=>"I
  love Rails"}
  assert assigns(:post).valid?
  assert_redirected_to post_path(assigns(:post))

  post '/posts/post_comment', :comment =>
  {:post_id=>(:post).id, :comment=>"my comment"}
  assert assigns(:comment).valid?
  assert_redirected_to :action => 'show'

  post '/posts/post_comment', :comment =>
  {:post_id=>(:post).id, :comment=>"my comment 2"}
  assert assigns(:comment).valid?
  assert_redirected_to :action => 'show'
end
```

RSpec

Thank you!

We do Instructor-led Codecamps!
http://www.javapassion.com/codecamps