Beginning Ruby on Rails

Session Five

Testing

- Depending on your experience, the idea of testing could be a little abstract...
  - We're writing code to test the code we wrote earlier?
  - Can't we just run the program and click everything to make sure it works?
  - Do we need to test our tests?

Why Test?

- Testing ensures that your program meets requirements and provides the desired functionality.
- With a good test suite in place, you can make major changes without worrying about breaking existing functionality.

What To Test?

- In a word: everything
- We will discuss testing models, views, controllers, and routes.
- We will test individual components and interactions between multiple controllers.

Testing in Rails

Built in from day one.

Rails Tests

- Rails tests are based on a Ruby library called Test::Unit
- Basic tests are automatically generated in the “test” directory.
- These are just a starting point, they don’t really test anything.

Rails Environments

- Rails provides 3 different environments for us to work in:
  - Development
  - Test
  - Production

Preparing to Test

- Make sure db/schema.rb is up to date

```
bundle exec rake db:migrate
```

- Create the test database

```
bundle exec rake db:test:load
```

Rake Tasks

- rake
  - rake test
    - rake test:units
    - rake test:functionals
    - rake test:integration

Fixtures

- Edit test/fixtures/posts.yml
- Sample data for tests.
- Written in a format called YAML
  - YAML Ain't Markup Language
- Automatically loaded and available to all test cases.

Fixtures

- id values are based on a hash of the fixture name and are therefore always the same
- Associations are created by name
- Fixtures are automatically deleted and reloaded before each test case
- You can look at the test log to see the queries being run for each test

Test Helper

- Edit test/test_helper.rb
- Holds the default configuration for tests.
- Loads all fixtures.
- Methods added to this file will be available in all tests.

Callbacks

- setup and teardown
- Special callback methods that run before and after each test case.
- Useful to assign variables or initialize data used in multiple tests.
- Remember DRY

Assertions

- The building blocks of tests

```
assert
assert_equal
assert_match
assert_difference
```

Unit Tests

Tests for your models.

Unit Tests

- Edit test/unit/post_test.rb
- Also reference app/models/post.rb
- We should test validations and any custom methods we have written.

```
rake test:units
```

Functional Tests

Tests for controllers and views.

Functional Tests

- Test the posts controller, views, and routes
- test/functional/posts_controller_test.rb
- Scaffolding generated some real tests.

```
rake test:functionals
```

Functional Helpers

- Request methods:
  - get, post, put, head, delete
- Hashes:
  - assigns, cookies, flash, session
- Instance variables:
  - @controller, @request, @response

Views and Routes

- Use assert_select to test views

```
assert_select ‘title’, “Tony’s Blog”
```

- Use assert_routing to test routes

```
assert_routing '/posts', { :controller =>
"posts", :action => "index" }
```

Integration Tests

Make sure everything fits together.

Integration Tests

- Used to test the interaction among any number of controllers.
- Generally used to test important work flows within your application.
- Must be manually created:

```
rails generate integration_test
add_comment
```

Integration Helpers

- In addition to the other test helpers:
  - get_via_redirect
  - post_via_redirect
  - put_via_redirect
  - delete_via_redirect

TDD

Test Driven Development

TDD

- Write a failing test
- Write code to make the test pass
- Refactor as needed
- Repeat

Testing Comments

- Let's use TDD to add validation to the Comments model.
  - Edit app/models/comment.rb
  - Edit test/unit/comment_test.rb
  - Edit test/fixtures/comments.yml

```
rake test:units
```

Other Gems

Everyone has a favorite.

RSpec

- A Behavior Driven Development framework for Ruby.
- Focuses on documentation and design.
- Tests are written more like specifications.
- http://relishapp.com/rspec

Cucumber

- Another tool for BDD
- Executes plain-text functional descriptions as automated tests.
- The language that Cucumber understands is called Gherkin.
- http://cukes.info

Capybara

- Capybara provides a Ruby DSL for interacting with web pages.
- Used to simplify integration testing
- Simulates a user interacting with your application.
- https://github.com/jnicklas/capybara

AutoTest

- Scans the files in your project for changes and runs the appropriate tests.
- Failures are repeated until they all pass.
- Then the full test suite is run to ensure that nothing else was inadvertently broken.
- http://zentest.rubyforge.org/ZenTest/Autotest.html

Guard

- Guard is a command line tool to easily handle events on file modification.
- Useful for more than just tests.
- Can also restart server, reload browser, etc.
- https://github.com/guard/guard

Homework