



RUBY (ON RAILS)

Content adapted from material by Ryan Tucker and Kelly Dunn
University of Washington, CSE 190M, Spring 2009

Announcements

2

- Return Quizes
- Second Test: Wednesday April 23
- Project Presentations
 - Monday, April 28
 - Wednesday, April 30

Scrum Masters

3

Backslash	MICHAEL	HOLUPKA
C.O.D.E.	MINGJIAN	ZHANG
Cellar	EVAN	BASTA
ContraWeb	RUBY	REYNOSO
Hacklemore	EMILY	ANSLEY
Lannister	EDWARD	SAMUALS
Llama	CHRISTOPHER	BELL
Sk3m Team	MATTHEW	NING
SqlThePrql	JEREMY	WARNER
Synapps	CHARLES	KELMAN
Tautology	TAIT	MADSEN
Team RNG	CHI MAN	WONG



Heartbleed

- A bug in openSSL, a common open source implementation of Secure Sockets Layer (SSL)
 - ▣ SSL provides the encryption for https.
 - ▣ The bug gives access to memory possibly passwords
- SSL uses public key encryption
 - ▣ ability to securely exchange encryption keys.
- Apache can be configured to use https
 - ▣ A good idea if you are exchanging passwords
 - ▣ Can redirect from http to https

Web Application Frameworks

- Java Web Software
- Microsoft Web Software
- Ruby on Rails

Java Web Software

- Integrated Development Environments (IDE)
 - NetBeans
 - Eclipse
- Java to program the web
 - Servlets (extends HttpServlet)
 - Methods (doGet, doPost, doPut, doDelete)
 - `PrintWriter out = response.getWriter()`
 - Servlet Containers
 - Apache Tomcat, GlassFish

Java Web Software (cont.)

▣ Java Server Pages (.jsp)

■ JSP Expression Language (EL)

- `${ expression }`

■ JSP Standard Tag Library (JSTL)

- `<% @ taglib prefix = "c" uri = "http://java.sun.com/jsp/jstl/core" %>`

- `<c:if test = "boolean expression"> JSP </c:if>`

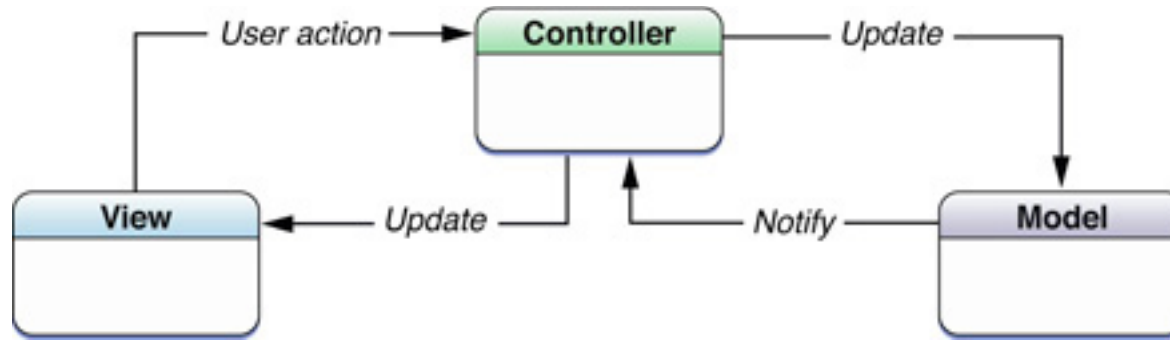
▣ Java Server Faces

■ Event driven user interface model

Microsoft (ASP.net)

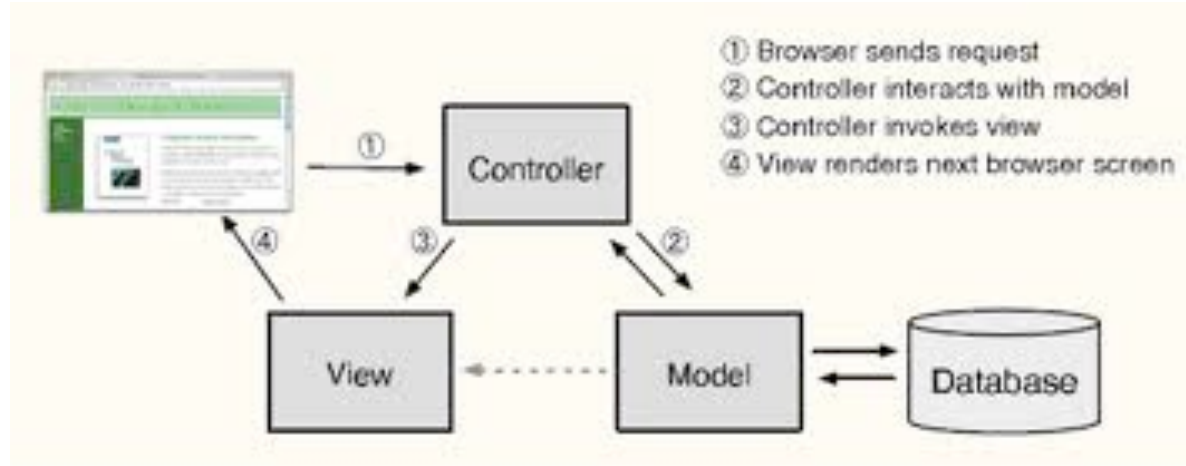
- Active Server Pages (ASP)
 - IDE: Visual Studio .Net
 - Language C#
 - ASP container: IIS
 - ASP.net controls (e.g. Button, Checkbox ...)
 - `<asp:textbox id = "age" runat = "server"/>`

Model View Controller



- Model: describes current state and valid changes
- View: displays current state and accepts user actions
- Controller: updates view and controller

Relationship to 3 level architecture



- View = UI = Client/Browser + Page construction
- Model = Database/Business Logic
- Controller = orchestrator of Model and View

Ruby on Rails

- Implemented in and uses Ruby
- Based on Model View Controller
- Example
 - ▣ rails new greet
 - creates subdirectories: models, views and controllers
 - ▣ rails generate controller say hello
 - creates controller for “say” with action “hello”
 - ▣ rails generate scaffold db item1:string item2:integer
 - creates database, model and view

12

Standup

Discuss questions with your Scrum Team

13

Quiz

Team Quiz (team name & members)

14

- Give an example of each of the three types of relationships between objects in a model.

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year><price>30.00</price>
  </book>
  <book category="computers">
    <title lang="en">XQuery Kick Start</title>
    <author>James McGovern</author>
    <year>2003</year><price>49.99</price>
  </book>
  <book category="computers">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year><price>39.95</price>
  </book>
</bookstore>
```

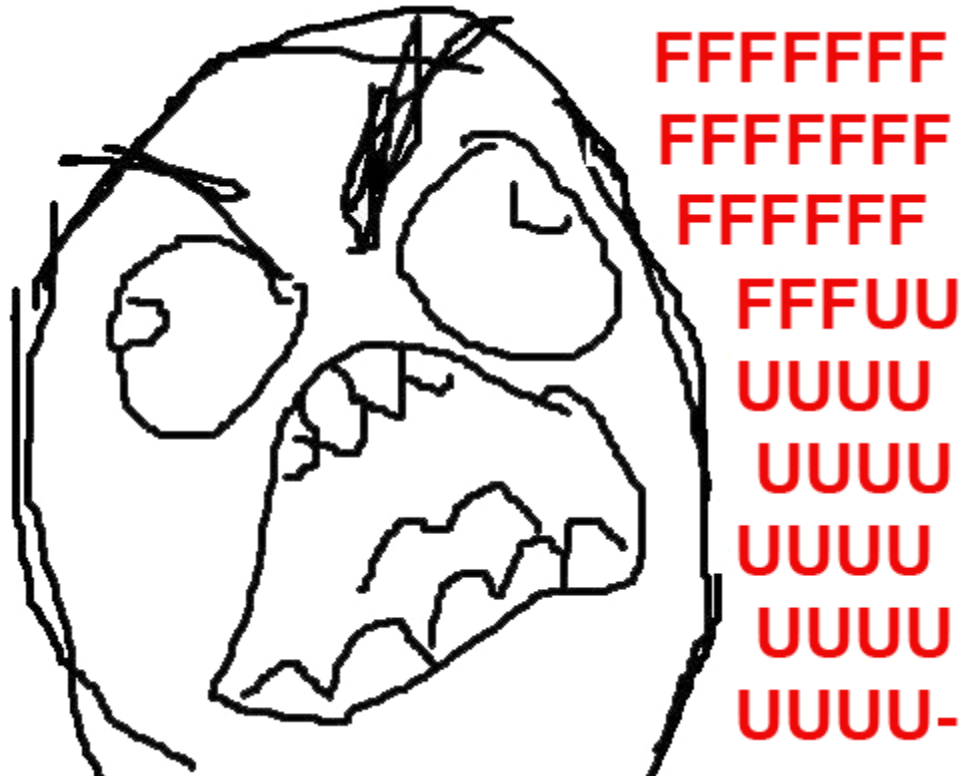
XML

Write two JavaScript functions:

1. A function that will make an Ajax request to `fubar.bookstore.com`
2. The function specified in your Ajax request, that will create a page consisting of the books in the “cooking” category listing the author and the title of the book.

Yet another language to learn!

- Inundated beginner's response:



But actually...

- Many languages look suspiciously similar
- Fact of life on the web



≡



(no, Ruby was not made with PHP)

Today (and Monday)

- Introduce the Ruby programming language
- Use Ruby to template web pages
- Learn about Ruby on Rails and its benefits



What is Ruby?

- Programming Language
 - ▣ General purpose
 - ▣ Relatively new (1995)
- Object-oriented
- Rare gem?

What is Ruby on Rails? (RoR)

- Development framework for web applications
 - ▣ Written in Ruby
- Notable RoR-based sites:
 - ▣ <http://rubyonrails.org/applications>

github
SOCIAL CODING

hulu



Popularity

- Hype has calmed, what's left?



Advantages of a framework

- Standard features/functionality are built-in
- Predictable application organization
 - ▣ Easier to maintain
 - ▣ Easier to get things going

hello_world.rb

```
puts 'hello world!'
```

Running Ruby Programs

- Use the Ruby interpreter
 - `ruby hello_world.rb`
- Interactive Ruby (irb) console
 - `irb`
 - ▣ Get immediate feedback
 - ▣ Test Ruby features

Ruby syntax in (5..10)

- Live demo! Including:
 - ▣ Comments, variables, objects, operators
 - ▣ Classes, methods, message passing
 - ▣ “Everything is an object”
 - ▣ Conditionals, loops
 - ▣ Arrays, ranges, strings, hashes, bears
- Q&A
 - ▣ Your syntax Qs, my syntax As

(no guarantees that my demo won't be on a chalkboard)

Blocks

- Unique enough to dive into detail
- Blocks are simply "blocks" of code
- Defined by `{ }` or a `do/end` statement
- Used to pass code to methods and loops

Blocks

- Many languages “only” have function args
- In Ruby, we can pass code through blocks
- Example: the `times()` method takes a block:
`3.times { puts "hello" }` # the block is the code in the `{ }`

Blocks and Arguments

- Blocks can also take arguments
- `times()` takes a block that takes an argument
- Example
 - `3.times { |n| puts "hello" + n.to_s }`
- "n" is specified as an argument via pipes (|)

Web Programming in Ruby

- Ruby can be used to write dynamic web pages (!)
 - ▣ With and without Rails
- Chunks of Ruby begin with `<%` and end with `%>`
 - ▣ Called *embedded Ruby*
- Ruby-based pages have file extensions of `.erb`
- Web servers need to be told to interpret `.erb`

erb syntax

- Code blocks
- Roughly equivalent to `<?php ... ?>` in PHP
 - `<%`
 - ruby statements***
 - `%>`
- Printing expression values
- Equivalent to `<? = ... ?>` in PHP
 - `<% = expression %>`

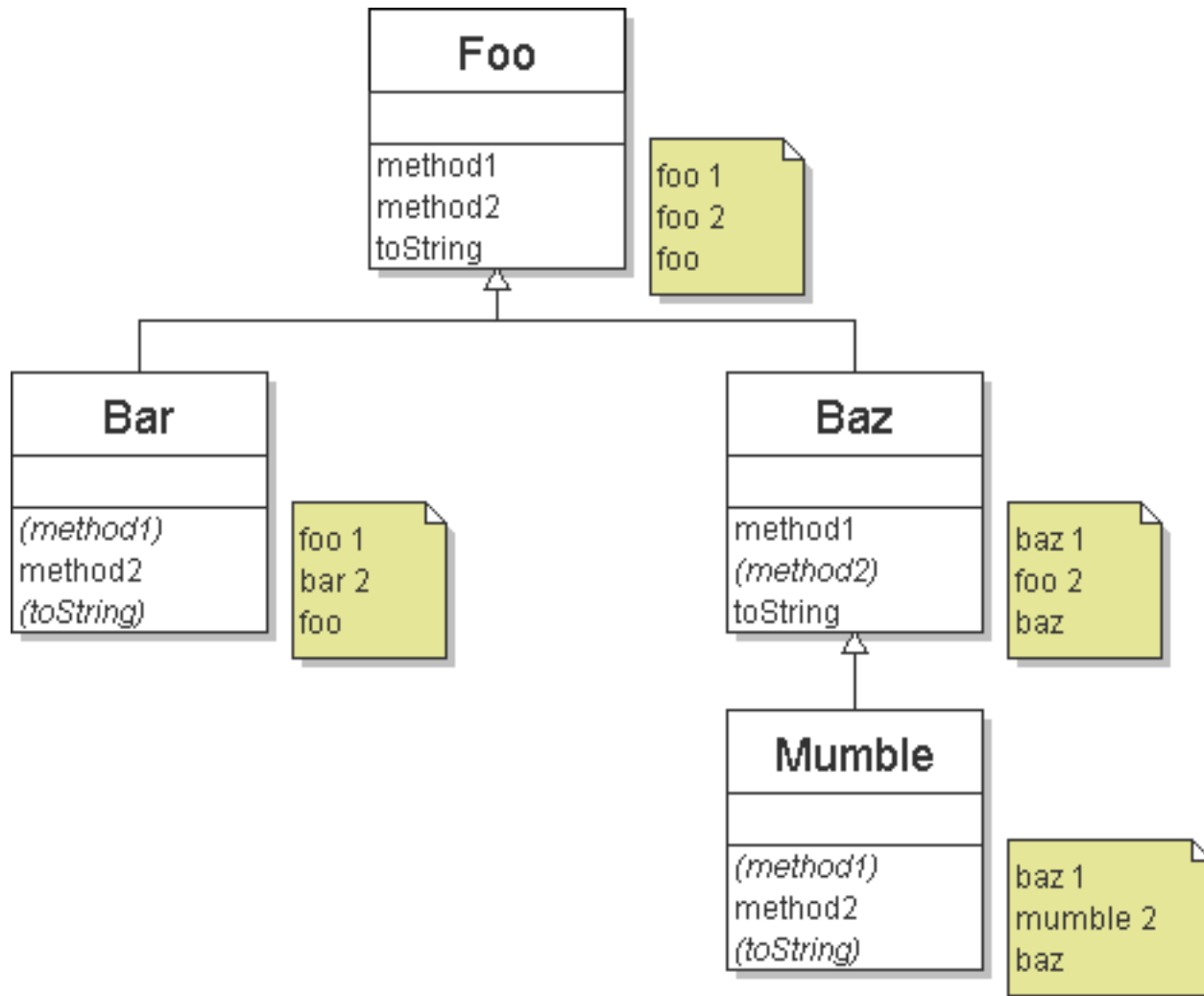
Example: 99 Bottles of Beer

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head><title>Embedded Ruby</title></head>
  <body>
    <%
      num = 99
      while num > 0
        %>
          <p> <%= num %> bottles of beer on the wall, <br />
            <%= num %> bottles of beer. <br />
            Take one down, pass it around, <br />
            <%= num - 1 %> bottles of beer on the wall. </p>
        <%
          num = num - 1
        end
      %>
    </body>
  </html>
```

Classes and inheritance

- Ruby supports single inheritance
- *Mixins* provide supplemental inheritance
 - ▣ A class can *extend* multiple *modules*
 - (in addition to the class inheritance chain)
 - ▣ Individual instances can extend them too

Inheritance



Modifying Class Behavior

- Add functionality to ANY class
 - ▣ “open” classes
 - ▣ still open after initial declaration

- Includes built-in classes!

RUBY ON RAILS



What is Ruby on Rails?

- Rails is...
 - ▣ Written in Ruby
 - ▣ A web development framework
 - ▣ For development of web apps written in Ruby
- Benefits of Rails
 - ▣ Built-in functionality
 - ▣ Encourages good software development practices
 - ▣ Open source and lots of community support

What is Ruby on Rails?



Disadvantages of Rails

- Assumes familiarity with Unix-like systems
 - ▣ cd – change directory
 - ▣ ls – list file and folders in current folder
- Lots of "magic"
 - ▣ Not always clear how things are being done
 - ▣ Not always obvious how to change/debug "magic"
- Deployment.....

Installing Rails

- It's already on Betaweb
 - ▣ 2.1.1... current version is 3.0.5 or so
 - ▣ BTW, Ruby version is 1.8.5... vs. 1.8.7 or 1.9.2
- Check the reading column in class schedule

Creating a New Rails App

- Give 'rails' executable the path to our new app
- Create your Rails application!
`rails -d mysql path/to/application`
- Example
 - `rails -d mysql my_app`
- Spits out a structured folder of files
 - These files form the base of all RoR apps

Starting Your Application

- Start Webrick, the built-in webserver
`ruby my_app/script/server`
- Amazingly, this works on Betaweb

Viewing Your Application

- Working locally

- <http://localhost:3000/>

- Working on Betaweb

- Avoid port collisions with fellow students

- `ruby my_app/script/server -p xxxx (x > 1023)`

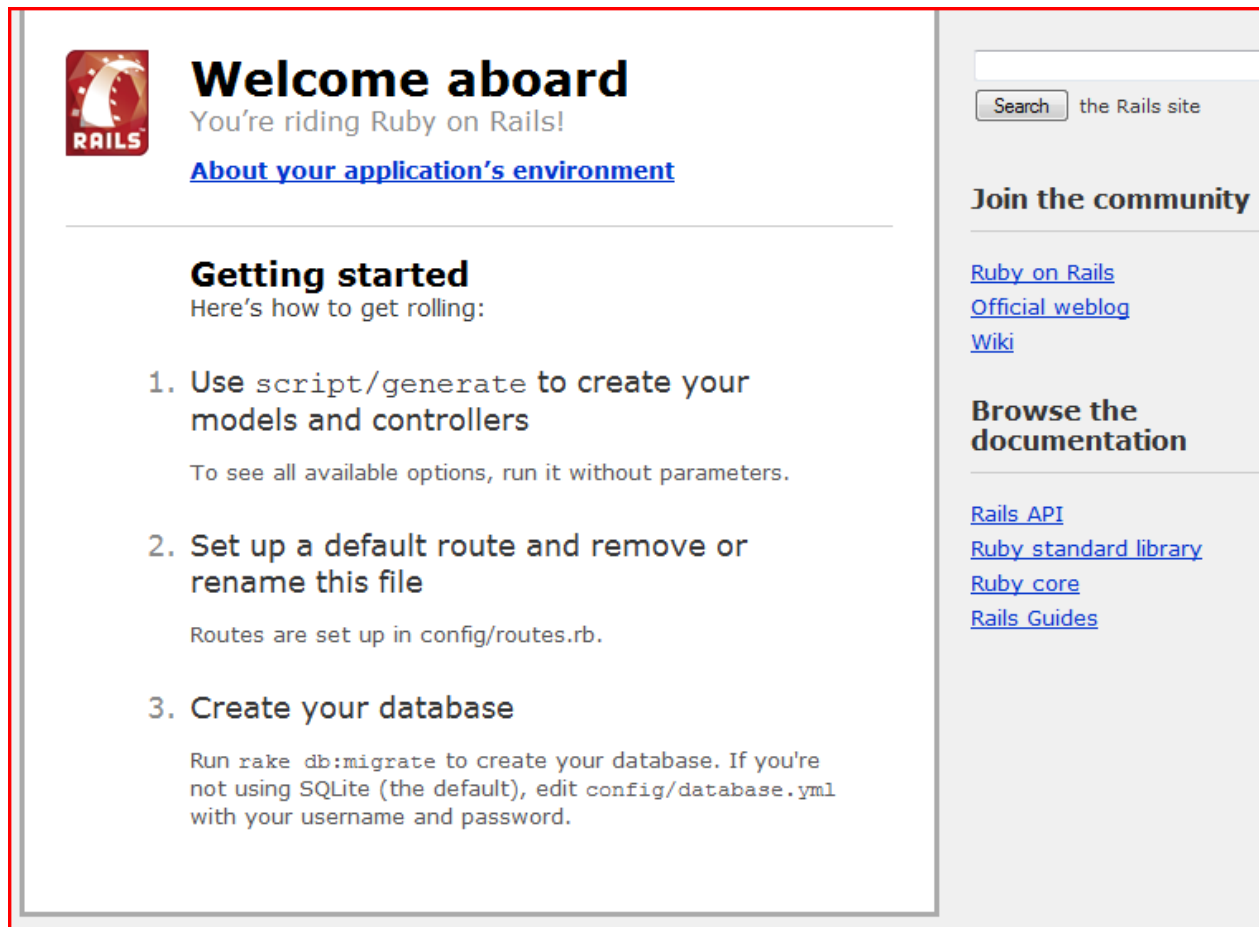
- Avoid CS firewall

- `ssh -Y user@cycle2.csug.rochester.edu`

- `firefox betaweb.csug.rochester.edu:xxxx &`

Viewing Your Application

- A new application will show you this page



The screenshot shows the 'Welcome aboard' page for a new Ruby on Rails application. It features the Rails logo, a search bar, and navigation links for community resources and documentation.

Welcome aboard
You're riding Ruby on Rails!
[About your application's environment](#)

Getting started
Here's how to get rolling:

1. Use `script/generate` to create your models and controllers
To see all available options, run it without parameters.
2. Set up a default route and remove or rename this file
Routes are set up in `config/routes.rb`.
3. Create your database
Run `rake db:migrate` to create your database. If you're not using SQLite (the default), edit `config/database.yml` with your username and password.

Join the community
[Ruby on Rails](#)
[Official weblog](#)
[Wiki](#)

Browse the documentation
[Rails API](#)
[Ruby standard library](#)
[Ruby core](#)
[Rails Guides](#)

Navigating the Rails File System

- A default app folder has a number of folders
- We are interested in only a few of them now
 - ▣ The "app" folder, specifically "app\views"
 - ▣ The "config" folder

The "app" Folder

- The "app" folder deals with the actual code of our application.
- It will hold all of our...
 - ▣ Objects ("**M**odels"),
 - ▣ .erb files ("**V**iews"), and...
 - ▣ code to work between the two ("**C**ontrollers")
- MVC pattern
 - ▣ Detailed later

The "config" Folder

- Unsurprisingly contains settings
- routes.rb controls URI mappings
- database.yml holds database connection info
 - Your Betaweb account info goes here
 - You only get one database on Betaweb =(

Databases with RoR

- When creating your Rails app...
 - ▣ Special flag to switch to MySQL
`rails -d mysql my_app`
- Modify `config/database.yml`
 - ▣ Username & database name are the same

Web Applications

- Consist of models, views, and controllers
 - ▣ Together, these deal with user page requests
 - ▣ Like Ruby, RoR represents most things as objects
- In a typical, dynamic web app:
 - ▣ Database records are objects (model).
 - ▣ Multiple ways to display models (views)
 - ▣ We want these to communicate (controllers)

Scaffold

- Coding MVC structure by hand can be tedious
- Rails has the ability to generate a “scaffold”
 - ▣ Skeleton code to fit our object specs
- *scaffold* generates the files automatically
 - ruby script/generate scaffold **Object field1:datatype field2:datatype**
 - e.g. ruby script/generate scaffold Entry title:string data:text

Scaffold (continued)

- Generated code follows a URI standard
 - ▣ List All (GET /entries) – shows all entries
 - ▣ Show (GET /entries/1) – shows details of a particular entry
 - ▣ Edit (GET /entries/1/edit) – edits a particular entry
 - ▣ Update (PUT /entries/1) – updates a particular entry
 - ▣ New (POST /entries) – creates a new entry
 - ▣ Delete (DELETE /entries/1) – deletes a particular entry

Scaffold (continued)

- Routing (URI) standardization occurs in the routes.rb file
 - ▣ Specifies which controller and action handles each type of request
- The code to deal with the requests are found in the controller's methods (index, show, create, delete, etc.)
- The page to be displayed has a file name corresponding to the action being used (index.html.erb, show.html.erb, etc.)

Using the Generated Code

- We can modify the models, views, and controllers
- The scaffold also generated code to create the necessary tables in our database (`my_app/db/migrate`).
- To actually run this code and update our database, run the following command in the application:

```
rake db:migrate
```
- Start our app and view the newly created scaffolding
`localhost:3000/entries`

Error Logging

- Anytime an error occurs, it is logged in the application
- You can find a stack trace of the errors in the application logs
 - `my_app/logs`

Adding Additional Views

- Scaffold generates a bunch of standard views and a template layout
- If we want additional actions we:
 - Add the action in our controller
 - Create the corresponding view
 - Update our routes.rb file
 - `map.resources :obj_plural => { :action => method }`
 - e.g. `map.resources :entries => { :preview => get }`

Templates with Layouts

- Scaffold creates a Layout that works as a template for our objects
`layouts/entries.html.erb`
- This template displays the same thing for each of the actions for that object, and then yields to the corresponding view to display the unique content
- If we want, we can make one single template to be used by all objects by making one named "layouts/application.html.erb"

Partial Layouts

- Sometimes we may want the same bit of code to show up in multiple templates (e.g. navigation bar)
- We can display partial pages in our templates or views, and they are called "partials"
- Partials can be in any of the views or layouts folders
- By convention, they start with an underscore
views/layouts/_top_nav.html.erb
- We can render our partial in a template wherever we want it to be displayed
`<%= render(:partial => "layouts/top_nav") %>`

Models

- If you have inspected the Models created by the Scaffolding, you will notice that they are empty
 - ▣ But they inherit from `ActiveRecord::Base`
 - ▣ This is what gives us access to the fields (methods) of the objects as they are in the database without defining the methods ourselves
- So... why do we even have models?

Models

- We use Models define methods that provide information about individual objects
- These methods usually do calculations on other properties of the object that you will not want to write over and over again

- Example:

<code>user.admin?</code>	<code># checks if a user is a admin user</code>
<code>user.authenticate</code>	<code># authenticates a user</code>
<code>gallery.empty?</code>	<code># checks if a gallery is empty</code>
<code>gallery.clear</code>	<code># removes all the gallery images</code>

Relationships

- Often, there are inherit relationships between the different object we are creating
 - In a blog
 - Users have many Entries; an Entry belongs to only one User
 - Entries have many Comments, and a Comment belongs to only one Entry
 - In a login system
 - Users have many Roles; Roles belong to many Users
 - In a course registration system
 - A Student has many courses; a course has many students

Types of Relationships

- One-to-One
 - ▣ A U.S. citizen has only one S.S.N; Each S.S.N. belongs to only one U.S. citizen
- One-to-Many
 - ▣ A person owns many cars; A car belongs to only one owner
 - ▣ A company has many employees; An employee is employed by only one company
- Many-to-Many
 - ▣ A student has many courses; A course has many students
 - ▣ A programmer has many projects; A project has many programmers
 - ▣ A blog post has many posters; A poster has many posts

Relationships in Models

- One-to-One
 - ▣ `has_one/belongs_to`
- One-to-Many (most common)
 - ▣ `has_many/belongs_to`
- Many-to-Many
 - ▣ `has_many/has_many`
 - ▣ `has_and_belongs_to_many`
 - ▣ These are tricky... So we will not go into detail with these
- An object that `belongs_to` another should reference that object by id in the database
 - `employee.company_id`

Relationships in Models

In Our Gallery Class

```
class Gallery < ActiveRecord::Base
  has_many      :images      # plural: images
end
```

In Our Image Class

```
class Image < ActiveRecord::Base
  belongs_to    :gallery     # singular: gallery
end
```

References

- Web Sites

- ▣ <http://www.ruby-lang.org/en/>

- ▣ <http://rubyonrails.org/>

- Books

- ▣ Programming Ruby: The Pragmatic Programmers' Guide
(<http://www.rubycentral.com/book/>)

- ▣ Agile Web Development with Rails

- ▣ Rails Recipes

- ▣ Advanced Rails Recipes