Ruby on Rails

CSE 190M, Spring 2009

Week 6

Overview

- How to use a database
- Demo creating a blog application on Rails
- Explain how the application works and how we can modify it
- Show you how to get your application up and running on webster
- Extend the homework #5 due date!

Using a Database

- Install MySQL (this is already installed on Webster)
- Install the "mysql" gem (also on Webster)

  gem install mysql
- Create your Rails app

  rails –d mysql my_app
- Modify config/database.yml to have the correct username and password for your database
- Open MySQL and create the database for your application

  mysql > create database my_app_development

Sample database.yml

- This sample database.yml example uses the same database for development, test, and production environments
- By default, each environment (prod, test, dev) have their own database so as not to interfere with each other
- This sample specifies a database with the name "my_app_db", with user "boot" and password "hackme"

```
# Sample database.yml
defaults: &defaults
  adapter: mysql
  database: my_app_db
  username: boot
  password: hackme
  host: localhost

development:
  <<: *defaults

production:
  <<: *defaults
```

Web Applications

- Last week, we saw how to create a static, custom homepage
- We had to create a Model, View, and Controller to deal with the request
- In the spirit of Ruby, Rails represents everything as objects
- In a large, dynamic web app, everything would be representing as an object (model). For each model, there may be many ways to display it (views), and we would want these to communicate (controllers)

Scaffold

- We could write a web app by hand, creating all models, views, and controllers by hand
- However, Rails recognizes that there is standard functionality that is done over and over again
- To avoid doing all of this by hand, Rails has the ability to generate a "scaffold", or skeleton code, to fit our object specs
- In the application root, use the scaffold to generate the files automatically

  ruby script/generate scaffold ***Object field1*:*datatype field2*:*datatype***

  e.g. ruby script/generate scaffold Entry title:string data:text

HTTP

- HTTP – hypertext transfer protocol
- Establishes a client-server relationship and details how they should communicate to exchange information
  - Client makes and HTTP request
  - Server returns a response
- We have already seen some HTTP request methods
  - GET, POST
- There are others
  - PUT, DELETE, etc.

Scaffold and HTTP

- The scaffolding generates code that corresponds to the HTTP request methods
- The corresponding code deals with the requests in a standardized way
  - 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 and HTTP

- Routing standardization occurs in the routes.rb file
- The routes.rb file specifies which controller and action (method) should handle each type of request

```
map.resources :entries
```

- 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 as we feel fit
- 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 (we do not have to touch the database ourselves)

  rake db:migrate

- Start our app and view the newly created scaffolding

  localhost:3000/entries

Rails on Webster

- In your public_html directory, make a folder for your Ruby apps

  /home/rctucker/ruby_apps

- Create your Rails app in this folder

  rails –d mysql my_app

- In your public_html folder, make a symlink from the public folder of your app to a folder with the name of your app

  ln -s ruby_apps/my_app/public my_app

- Create/modify .htaccess file in your public_html folder. Add the following line (using your username and application name instead)

  RailsBaseURI /rctucker/my_app

- View app at webster.cs.washington.edu/***username***/***app_name***

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