CSCI-2320
Web Programming: Ruby on Rails
Mohammad T. Irfan

Plan

- Model-View-Controller (MVC) framework of web programming
- Ruby on Rails
Ruby on Rails

- Developed by David Hansson - released 2004
- MVC architecture
  - MVC by Trygve Reenskaug, 1979
  - GUI for Smalltalk
- Learning Resources
  - Quick guide
    http://guides.rubyonrails.org/getting_started.html
  - Best online book
    https://www.railstutorial.org/book

Interview of David H. Hansson

- “Ruby Is Closer to Human Thought Than to Code”
  - https://bigthink.com/u/davidheinemeierhansson
Ruby on Rails - MVC framework

- **Goal**
  - Decouple the three parts of an application

- **Model**
  - Database
  - Constraints on data
  - Object Relational Mapping (ORM)
    - Maps tables to classes, rows to objects
    - Called ActiveRecord

Ruby on Rails - MVC framework

- **View**
  - Prepares and presents results for users
  - Templates
    - XHTML
    - XML
    - Javascript
Ruby on Rails - MVC framework

- Controller
  - Takes user input
  - Consults with model
  - Directs the view
  - The basic codes are auto-generated

Getting started

- Command
  - `rails new projectName`
  - Windows users: open Gemlock.lock file, change "sqlite 3 (1.3.8-...)" to "sqlite 3 (1.3.8)"

- Error related to Gemfile?
  - cd to project folder
  - Execute command: `bundle install`

- Start the server
  - cd to project folder
  - Execute command: `rails server`

- Open a browser and go to [http://localhost:3000/](http://localhost:3000/)
Browse the project folders

- App
  - models
  - views
  - controllers

Scaffolding

- Fast process of generating start-up codes
- First, design a schema

<table>
<thead>
<tr>
<th>bID</th>
<th>name</th>
<th>email</th>
</tr>
</thead>
<tbody>
<tr>
<td>B01224</td>
<td>Bob</td>
<td><a href="mailto:bob@bowdoin.edu">bob@bowdoin.edu</a></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

- Command for ORM
  - rails generate scaffold Student bID:string name:string email:string
  - Other useful commands: rails destroy scaffold ... (delete a previous ORM)
Migrate model to DB

- Command for migration
  - `bundle exec rake db:migrate`
  - Reverse is: `rake db:rollback` (don’t run it now)
- “rake”
  - Ruby’s make: configure, make, make install
- “bundle exec”: executes the rake script (db:migrate) in the context of the project’s Gemfile

View the webpage

- Command
  - `rails s`
    - s is shortcut for server
- Go to [http://localhost:3000](http://localhost:3000) on web browser
  - No surprise there
Surprise!

Navigate to http://localhost:3000/students

What’s going on?

2. <Ruby Router> routes to students_controller.rb
3. students_controller.rb gets data from database table students
4. students_controller.rb feeds data to View<index.html.erb> within the students view (erb = embedded Ruby)
5. index.html.erb produces a nice html file and gives it to students_controller.rb
6. students_controller.rb gives that html file to Chrome
Rails router

- config/routes.rb
  - resources: students
- Routes to
  - app/controllers/students_controller.rb
- class StudentsController < ApplicationController
  ...
  # GET /students/new
  def new
    @student = Student.new
  end
  ...
end

Rails architecture

- Representational State Transfer (REST)
  - Roy Fielding (2000) - “architectural style”
- Clients communicate with web service
  - Limited number of verbs
  - Resources (nouns) - identified by URI
- Rails
  - Nouns: objects (tables) in ORM
  - Verbs: Read, create, update, delete
- HTTP
  - Nouns: URL
  - Verbs: GET, POST, PATCH, DELETE
Creating a new website

1. Create its own controller
2. Add pages to it later on

Website with dedicated controller

- Command
  - rails generate controller MyHomePage home contact --no-test-framework
- Controller class
- Views
Navigating to my_home_page/home...

- The home method of the Controller class is executed first
  - Empty for now
- Then the corresponding view is executed
  - home.html.erb
  - You may edit it as you like

Adding a page without adding new controller

- First, modify config/routes.rb
  - get “my_home_page/projects”
- Modify the controller class in my_home_page_controller.rb
  ```ruby
def projects
end
```
- Create view
  - Add a new projects.html.erb file in views/my_home_page folder
- Any content:
  ```html
  <h1>Here are my Ruby projects</h1>
  <%= image_tag 'Ruby_logo.png' %>
  ```
  Put it in app/assets/images
  More here:
  http://guides.rubyonrails.org/layouts_and_rendering.html
Test the pages

- Browse to

- Error - why?

Building an auction app from scratch

without scaffolding
Plan

- Rails web application
  - A more involved example
  - Without scaffolding
  - Understand flow of control
- Problem: web service with database connectivity - auction
  - Input: name and bid amount
  - Store bid information in database
  - Output: show all bids in sorted order

Welcome to the auction!

Google this picture
Download it (Copy to app/assets/images)

<table>
<thead>
<tr>
<th>Bidder</th>
<th>Bid</th>
</tr>
</thead>
<tbody>
<tr>
<td>David Kerrigan</td>
<td>250.0</td>
</tr>
<tr>
<td>Alice Lin</td>
<td>200.0</td>
</tr>
<tr>
<td>Rob Johnson</td>
<td>100.0</td>
</tr>
<tr>
<td>David Parkes</td>
<td>50.0</td>
</tr>
<tr>
<td>Bob Mitchell</td>
<td>25.0</td>
</tr>
<tr>
<td>Just Kidding</td>
<td>0.5</td>
</tr>
</tbody>
</table>
First Step

- Create an application
  - rails new AuctionApp
- `cd` to the project folder AuctionApp
- Create a controller - the only controller
  - rails generate controller AuctionApp index

Routing - `config/routes.rb`

- Make the index page the root (http://localhost:3000)
  - root "auction_app#index"
- Other routing information (previously these were done automatically when you said `resources :students`)
  - get "/auction_app" => "auction_app#index"
  - get "/" => "auction_app#index"
  - post "/" => "auction_app#enterBid"

```ruby
Rails.application.routes.draw do
  root "auction_app#index"
  get "auction_app" => "auction_app#index"
  get "/" => "auction_app#index"
  post "/" => "auction_app#enterBid"
end
```
Start the server

- Open 2 terminals- one for server, one for other commands
- In both terminals, you must \texttt{cd} to the appropriate project folder in terminal—in my case it’s the \texttt{AuctionApp} folder
  - Command to start server: \texttt{rails s}

Model - without scaffolding

- Create a model: ORM
  - \texttt{rails generate model Bid bidder:string amount:float}
- Create actual database table
  - \texttt{bundle exec rake db:migrate} # creates DB table \texttt{bids}
- Suppose we don’t want a separate controller for this (want to use \texttt{auction_app_controller})
  - Don’t say \texttt{resources :bids} in \texttt{routes.rb}
  - If you say so, it will automatically (without writing it explicitly in \texttt{routes.rb}) map HTTP get, post, etc. to \texttt{index}, \texttt{create}, etc. methods of the \texttt{bids_controller.rb} which we don’t have!
Controller

Action for the “Enter Bid” button
  auction_app/enterBid: enterBid method in auction_app_controller

Next: write this method
  This is the method that will be called when the “submit” button is pressed
  You are allowed to pick any name for the method
    Must match with the router though!

```ruby
class AuctionAppController < ApplicationController
  protect_from_forgery with: :null_session

  def index
    @allBids = Bid.all
    puts "# of bids = #{@allBids.size}"
    @allBids = @allBids.sort_by { |bid| [-bid.amount, bid.bidder] }
  end

  def enterBid
    bidder = params[:bidderInput]
    amount = params[:amountInput].to_f
    map = {"bidder" => bidder, "amount" => amount}
    newRow = Bid.new(map)
    respond_to do |format|
      if newRow.save
        puts "Success!"
        format.html{redirect_to auction_app_url} # Can create an error page
      else
        format.html{redirect_to "/"}
      end
    end
  end
end
```

Other DB functions

- newRow.save
- newRow.update
- newRow.destroy
- Bid.find(map)

View

```html
<h1>Welcome to the auction!</h1>
<p><%= image_tag "starry.jpg" %></p>
<!-- equivalent html code
<form name="bidInput" action="/" method="post">
  <div>
    <p>Your Name: <input type="text" name="bidderInput"></p>
    <p>Your Bid: <input type="text" name="amountInput"></p>
    <input type="submit" value="Enter bid"></p>
  </div>
</form>
<!--
```
To see the actual database files:
1. cd to db folder
2. command:
   sqlite3 development.sqlite3
   .tables
   .select * from bids;
Flow of control

- localhost:3000
  - routes.rb routes it to auction_app_controller’s index method
  - shows output of index.html.erb

- Enter data in form and press “Enter Bid” button
  - routes.rb routes it to auction_app_controller’s enterBid method (why not the index method?)
  - Redirects to homepage

Assignment on Rails
Group project (2 or 3 students)

- Create a Ruby-on-Rails project
- Open-ended
- Most basic requirements
  - Take user input
  - Process that input
  - Work with database
  - Show some output
  - Work with non-textual data
  - Use other gems
- Submit code and screen shots
  - One submission per group. Clearly mention the names of group members on Blackboard.
Multiple Forms (One Controller)

Auction App

- Create new button to find the leader
  1. View: add embedded Ruby (erb) code for new form [alternative: HTML]
  2. routes.rb: Enter the name of a new method to handle multiple posts
     - One post for entering bids
     - Another for finding leader
  3. Controller: New post-handler method and new method for finding the leader
View (index.html.erb)

```erb
  <!-- Another form on the same page:
  Find the leader's name -->
  <%= form_tag do %
    <p>Who's leading the auction now?</p>
    <%= submit_tag "Get Leader" %>
  <% end %>
```

routes.rb

```ruby
Rails.application.routes.draw do
  root "auction_app#index"
  get "/auction_app" => "auction_app#index"
  get "/" => "auction_app#index"
  post "/" => "auction_app#enterBid"
  post "/" => "auction_app#handlePost"
end
```

Next: add methods to the controller class
```ruby
26. def getLeader
27.   puts "-------------- In Get Leader ---------------"
28.   # Need to sort again, because every request creates
29.   # a new instance of Controller class (why?)
30.   @allBids = Bid.all
31.   @allBids = @allBids.sort_by { |bid| [-bid.amount, bid.bidder] }
32.   puts "Leader: #{@allBids[0].bidder}"
33.   respond_to do |format|
34.     format.html {redirect_to auction_app_url}
35.   end
36. end
37.
38. def handlePost
39.   if params[:commit] == "Enter Bid"
40.     enterBid
41.   elsif params[:commit] == "Get Leader"
42.     getleader
43.   end
44. end
45. end  # end of class AuctionAppController
```

Welcome to the auction!

Your Name: 
Your Bid: $ 

Enter Bid

<table>
<thead>
<tr>
<th>Bidder</th>
<th>Bid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
<td>200.0</td>
</tr>
<tr>
<td>Alice</td>
<td>100.0</td>
</tr>
<tr>
<td>David</td>
<td>50.0</td>
</tr>
<tr>
<td>Clint</td>
<td>25.0</td>
</tr>
</tbody>
</table>

Who's leading the auction now?  getLeader
Message from the Rails Server: note how post is handled

Started POST "/" for ::1 at
Processing by AuctionAppController#handlePost as HTML
  Parameters: {"utf8"=>"✓", "authenticity_token"=>"PObsUzwMxuw0/28xwUISw+Ny3NfkMgV4JVy/yzqxLVQgoPjoKaZbvorKSOw==", "commit"=>"Get Leader"}
---------- In Get Leader ----------
  Bid Load (0.2ms) SELECT "bids".* FROM "bids"
Leader: Bob
Redirected to http://localhost:3000/auction_app
Completed 302 Found in 6ms (ActiveRecord: 0.4ms)