GEDİZ UNIVERSITY

Computer Engineering Department

COM401
Software Engineering Laboratory

November 04, 2014

LAB-3:

Rails Introduction

Time: 2 lab hours

Objectives:

Practice with

- Ruby Symbols
- Routes
- MVC pattern
- CRUD operations
- Forms

Learn how to

- run **rottenpotatoes** application on saasbook virtual machine
- build your own Rails application : **myrottenpotatoes**

Lab Outcomes:

Practice with RubyMine IDE and Rails basics.

Exercise:

A- Run and examine "*rottenpotatoes*" application

In this part, we will run and examine rottenpotatoes application on saasbook image. We assume that you have already installed VirtualBox and downloaded the saasbook image. To use your virtual machine, follow the steps.

1. Run VirtualBox and click the New button to create a new VM.
2. When the VM Wizard appears, select the following options:
    - operating system: Linux
    - version: Ubuntu
    - RAM base memory: at least 1024 MB
    - Select "Use existing hard disk" and choose the .vdi file you downloaded in step 2.[1]

After starting the machine, we can make rottenpotatoes application run by following the steps:

a) Open a terminal window, then write

cd Documents

cd rottenpotatoes

[1] Run the VMimage , http://www.saasbook.info/bookware-vm-instructions

b) In this folder, start rails server by writing to the terminal window

rails start (or rails s)

c) Then open a browser and enter the address:

http://localhost:3000/movies

d) Examine the folder structure by considering MVC pattern.

File/Folder	Purpose
app/	Contains the controllers, models, views, helpers, mailers and assets for your application.
bin/	Contains the rails script that starts your app and can contain other scripts you use to deploy or run your application.
config/	Configure your application's runtime rules, routes, database, and more.
config.ru	Rack configuration for Rack based servers used to start the application.
db/	Contains your current database schema, as well as the database migrations.
Gemfile Gemfile.lock	These files allow you to specify what gem dependencies are needed for your Rails application. These files are used by the Bundler gem.
lib/	Extended modules for your application.
log/	Application log files.
public/	The only folder seen to the world as-is. Contains the static files and compiled assets.
Rakefile	This file locates and loads tasks that can be run from the command line. The task definitions are defined throughout the components of Rails. Rather than changing Rakefile, you should add your own tasks by adding files to the lib/tasks directory of your application.
README.rdoc	This is a brief instruction manual for your application. You should edit this file to tell others what your application does, how to set it up, and so on.
test/	Unit tests, fixtures, and other test apparatus.
tmp/	Temporary files (like cache, pid and session files)

File/Folder	Purpose
vendor/	A place for all third-party code. In a typical Rails application, this includes Ruby Gems and the Rails source code (if you optionally install it into your project).

[2]

List of routes for the application can be seen below.

```
saasbook@saasbook:~/Documents/rottenpotatoes$ bundle exec rake routes
    movies GET    /movies(.:format)          {:action=>"index", :controller=>"movies"}
           POST   /movies(.:format)          {:action=>"create", :controller=>"movies"}
 new_movie GET    /movies/new(.:format)      {:action=>"new", :controller=>"movies"}
edit_movie GET    /movies/:id/edit(.:format) {:action=>"edit", :controller=>"movies"}
     movie GET    /movies/:id(.:format)      {:action=>"show", :controller=>"movies"}
           PUT    /movies/:id(.:format)      {:action=>"update", :controller=>"movies"}
           DELETE /movies/:id(.:format)      {:action=>"destroy", :controller=>"movies"}
saasbook@saasbook:~/Documents/rottenpotatoes$
```

B- Build your own rottenpotatoes application : "*myrottenpotatoes*"

It is easy to create a Rails application including CRUD operations in by following steps below via RubyMine IDE.

Step 1- Create a new Rails application

(Note that it is possible to create a Rails project from command line by writing

rails new myrottenpotatoes)

This step leads to creation of a typical Rails application folder structure.

Step 2- Edit myrottenpotatoes/config/routes.rb by adding **movies** resource.

resource route (maps HTTP verbs to controller actions automatically):

resources :movies

Step 3- List all routes for this resource

Tools/ Run Rake Task.. / routes

(Note that it is possible to do it from command line by writing **rake routes**)

[2] http://guides.rubyonrails.org/v4.0.8/getting_started.html

You should see a list similar to the below.

```
    Prefix Verb   URI Pattern                Controller#Action
    movies GET    /movies(.:format)          movies#index
           POST   /movies(.:format)          movies#create
 new_movie GET    /movies/new(.:format)      movies#new
edit_movie GET    /movies/:id/edit(.:format) movies#edit
     movie GET    /movies/:id(.:format)      movies#show
           PATCH  /movies/:id(.:format)      movies#update
           PUT    /movies/:id(.:format)      movies#update
           DELETE /movies/:id(.:format)      movies#destroy
```

Step 4- Create a Movie model

Tools / Run Rails Generator .. / model /

Movie title:string rating:string description:text release_date:datetime

After this step, active record is invoked and two ruby file is created two ruby files shown in the figure below.

```
   invoke  active_record
   create    db/migrate/20141103204748_create_movies.rb
   create    app/models/movie.rb
```

Step 5 – Create Movie Table

After preparation of Movie schema, we can create the Movie table.

Tools / Run Rake Tasks.. / db:migrate

After migration command, the table is created on the db , and you should see an output similar to the below.

```
==  CreateMovies: migrating ====================================================
-- create_table(:movies)
   -> 0.0040s
==  CreateMovies: migrated (0.0040s) ===========================================
```

You can use **sqliteman** to open this sqlite3 db.

Step 6 – Add some test data into the Movies Table

First, open a rails console : **Tools / Run Rails Console…**

Copy-paste the following lines on the console.

```
#create a sample movies array

more_movies = [
    {:title => 'Aladdin', :rating => 'G',
     :release_date => '25-Nov-1992'},
    {:title => 'When Harry Met Sally', :rating => 'R',
     :release_date => '21-Jul-1989'},
    {:title => 'The Help', :rating => 'PG-13',
     :release_date => '10-Aug-2011'},
    {:title => 'Raiders of the Lost Ark', :rating => 'PG',
     :release_date => '12-Jun-1981'}
]


Movie.send(:attr_accessible, :title, :rating, :release_date)

more_movies.each do |movie|
   Movie.create!(movie)
end
```

Step 7- Create Movies controller

Tools/Run Rails Generators/ Controller/ movies

This operation leads to creation of the following folders and files.

```
create  app/controllers/movies_controller.rb
invoke  erb
create    app/views/movies
invoke  helper
create    app/helpers/movies_helper.rb
invoke  assets
invoke    coffee
create      app/assets/javascripts/movies.js.coffee
invoke    scss
create      app/assets/stylesheets/movies.css.scss
```

Step 8 - Add an **index method** the the movies_controller.rb to obtaining all movie records from database. (remember routes !)

```
def index
    @movies = Movie.all
end
```

Step 9 – Create a view(index.html.erb or index.html.haml) to match with this index method (remember : convention over configuration..)

```
<!DOCTYPE html>
<html>
```

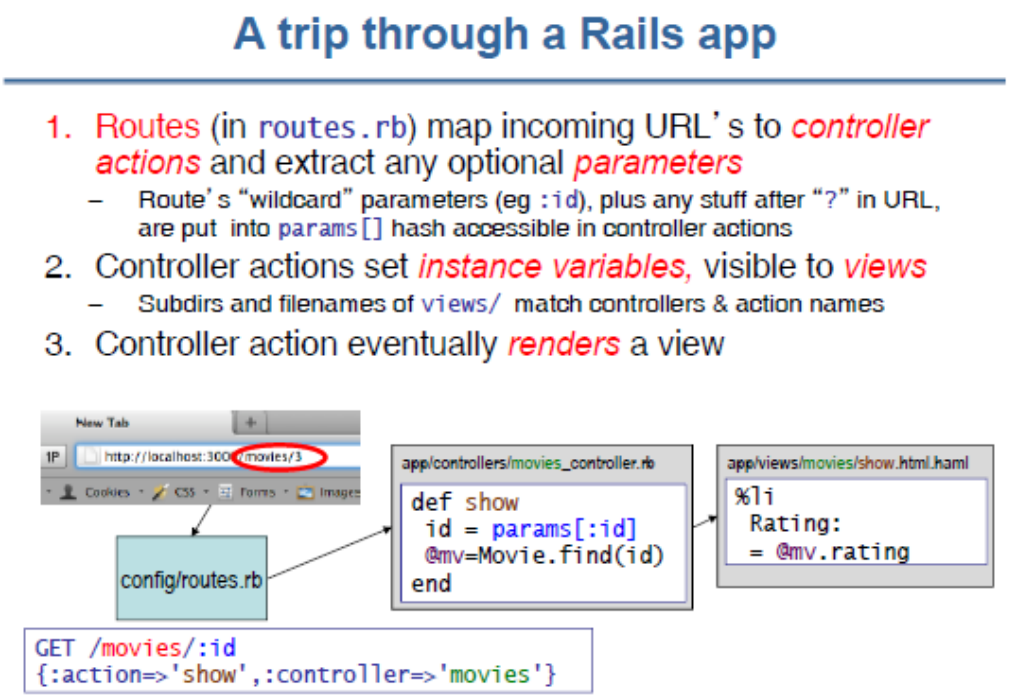
```
<body>
 <h2>All Movies</h2>
 <table>
     <thead>
       <tr>
         <td>  Movie Title     </td>
         <td>  Rating </td>
         <td>  Release Date </td>
         <td>  More Info </td>
       </tr>
     </thead>
     <tbody>
     <% @movies.each do |movie| %>
         <tr>
           <td>  <%= movie.title %>   </td>
           <td>  <%= movie.rating %>  </td>
           <td>  <%= movie.release_date %> </td>
           <td>  More about <a href= "movies/<%= movie.id %>">
                 <%= movie.title %></a>  </td>
         </tr>
         <%end%>

     </tbody>
  </table>
 </body>
</html>
```

Step 10 – Start your server(WEBrick) and test your web application in your browser

http://localhost:3000/movies

Just a remainder :



Step 11 - Add an **show method** the the movies_controller.rb to obtaining a specific movie record from database. (remember routes !)

```
def show
  id = params[:id] # retrieve movie ID from URI route
  @movie = Movie.find(id) # look up movie by unique ID
end
```

Step 12 - Create a new view(show.html.erb or index.html.haml) to match with this show method (remember : convention over configuration..)

```
<h2>Details about <%= @movie.title %> </h2>
<ul> details
  <li> Rating: <%= @movie.rating %> </li>
  <li> Released on: <%= @movie.release_date.strftime("%B %d, %Y")%></li>
</ul>

<h3> Description: </h3>
<p> description <%= @movie.description %>  </p>

<a href="<%= movies_path %>">Back to movie list</a>
```

Step 13 - Let's allow the users to add new movies

- First, add a new link whose title "Add a new Movie"

  Add the following line before closing body tag in the **index.html.erb** file

  ```
  <p><a href="movies/new"> Add a new Movie </a></p>
  ```

- Add a **new method** to the movies_controller.rb

  ```
  def new

  end
  ```

- Create a new view(new.html.erb or new.html.haml) to match with this new method (remember : convention over configuration..)

-

  ```
  <h2> Create New Movie   </h2>

  <%= form_tag movies_path, :method => :post do %>
      <%= label :movie, :title, 'Title' %>
      <%= text_field :movie, :title %>
      <%= label :movie, :rating, 'Rating' %>
      <%= select :movie, :rating, ['G','PG','PG-13','R','NC-17'] %>
      <%= label :movie, :release_date, 'Released On' %>
      <%= date_select :movie, :release_date %>
      <%= submit_tag 'Save Changes' %>
  <% end %>
  ```

- Add a **create method** the the movies_controller.rb to save the movie to the database

  ```
  def create
  ```

```
    @movie = Movie.new(movie_params)
    @movie.save
    flash[:notice] = "#{@movie.title} was successfully created."
    redirect_to movies_path
  end

  private
  def movie_params
    params.require(:movie).permit(:title, :rating, :release_date)
  end
```

- Add these lines inside the body tag of **app/views/layout/application.html.erb**

```
<%= yield %>
<% if flash[:notice] %>
<div class="message" id="notice">
<%= flash[:notice] %>
</div>
<% elsif flash[:warning] %>
<div class="message" id="warning">
<%= flash[:warning] %>
</div>
<% end %>
```

Step 14 - Let's allow the users to update and delete movies

- First, add a new links whose title "Edit info" and "Delete Movie"

  Add the following line before **"`Back to movie list`"** link in the **show.html.erb** file

```
<%= link_to 'Edit info', edit_movie_path(@movie) %>
<% # This Delete link will not really be a link, but a form: %>
<%= link_to 'Delete', movie_path(@movie), :method => :delete %>
```

- Add **edit, update** and **destroy methods** to the movies_controller.rb

```
def edit
    @movie = Movie.find params[:id]
  end

  def update
    @movie = Movie.find params[:id]
    @movie.update!(movie_params)
    flash[:notice] = "#{@movie.title} was successfully updated."
    redirect_to movie_path(@movie)
  end

  def destroy
    @movie = Movie.find(params[:id])
    @movie.destroy
    flash[:notice] = "Movie '#{@movie.title}' deleted."
    redirect_to movies_path
  end
```

- Create a new view(**edit.html.erb** or edit.html.haml) to match with these methods

  (remember : convention over configuration..)

```
<h2>Edit Movie</h2>
<%= form_tag movie_path(@movie), :method => :put do %>
    <%= label :movie, :title, 'Title' %>
```

```
    <%= text_field :movie, 'title' %>
    <%= label :movie, :rating, 'Rating' %>
    <%= select :movie, :rating, ['G','PG','PG-13','R','NC-17'] %>
    <%= label :movie, :release_date, 'Released On' %>
    <%= date_select :movie, :release_date %>
    <%= submit_tag 'Save Changes' %>
<% end %>
```

Step 15 - Provide a better web pages using style sheets.