Ruby On Rails

CSCI 5449

Submitted by: Bhaskar Vaish

What is Ruby on Rails ?

Ruby on Rails is a web application framework written in Ruby, a dynamic programming language.

Ruby on Rails uses the Model-View-Controller (MVC) architecture pattern to organize application programming.

What is Ruby on Rails ?(Continued)

- A *model* in a Ruby on Rails framework maps to a table in a database
- A controller is the component of Rails that responds to external requests from the web server to the application, and responds to the external request by determining which view file to render
- A view in the default configuration of Rails is an erb file. It is typically converted to output html at run-time

Ruby

- It is a dynamic, general-purpose **object-oriented programming** language
- Combines syntax inspired by Perl, also influenced by Eiffel and Lisp
- Supports multiple programming paradigms, including functional, object oriented, imperative and reflective

Ruby(Contd.)

- Has a dynamic type system and automatic memory management
- Ruby is a metaprogramming language.

Sample Ruby Code

# Output “Bhaskar”

```
puts “Bhaskar”
```

# Output “Bhaskar” in upprecase

```
 puts “Bhaskar”.upcase
```

# Output “Bhaskar” 10 times

```
10.times do
puts “Bhaskar”.upcase
 end
```

Sample Ruby Code: Class

Class Employee: defining three attributes for a Employee; name, age, position

```
class Employee # must be capitalized
 attr_accessor :name, :age, :position
# The initialize method is the constructor
 def initialize(name, age, position)
   @name = name
   @age = type
   @position = color
end
```

New Employee

Creating an instance of the Employee class:

```
a = Employee.new(“JAY", “23", “Test Engineer")

b = Employee.new(“SAM", “24",  “Test Engineer")
```

Method

To be able to describe employees, we add a method to the employee class:

```
def describe
  @name + " is  of " + @age + " years"
   +" working as "
   + @position+ ".\n"
end
```

Calling Method

To get the description of Employee, we can call Employee with the describe method attached :

```
emp= a.describe
puts emp

or:
puts a.describe
```

Rails

- Rails is an open source Ruby framework for developing database-backed web applications

- The Rails framework was extracted from real-world web applications. Thus it is an easy to use and cohesive framework that's rich in functionality

Rails(Contd.)

- All layers in Rails are built to work together and uses a single language from top to bottom
- Everything in Rails (templates to control flow to business logic) is written in Ruby, except for configuration files - YAML

What is so special about Rails

- Other frameworks use extensive code generation, which gives users a one-time productivity boost but little else, and customization scripts let the user add customization code in only a small number of carefully selected points
  - Metaprogramming replaces these two primitive techniques and eliminates their disadvantages.
  - Ruby is one of the best languages for metaprogramming, and Rails uses this capability well.

What is so special about Rails

Scaffolding

- You often create temporary code in the early stages of development to help get an application up quickly and see how major components work together. Rails automatically creates much of the scaffolding you'll need.

What is so special about Rails

Convention over configuration

- Most Web development frameworks for .NET or Java forces to write pages of configuration code, instead Rails doesn't need much configuration. The total configuration code can be reduced by a factor of five or more over similar Java frameworks just by following common conventions.
    - Naming your data model class with the same name as the corresponding database table
    - ‘id’ as the primary key name

What is so special about Rails

Active Record framework

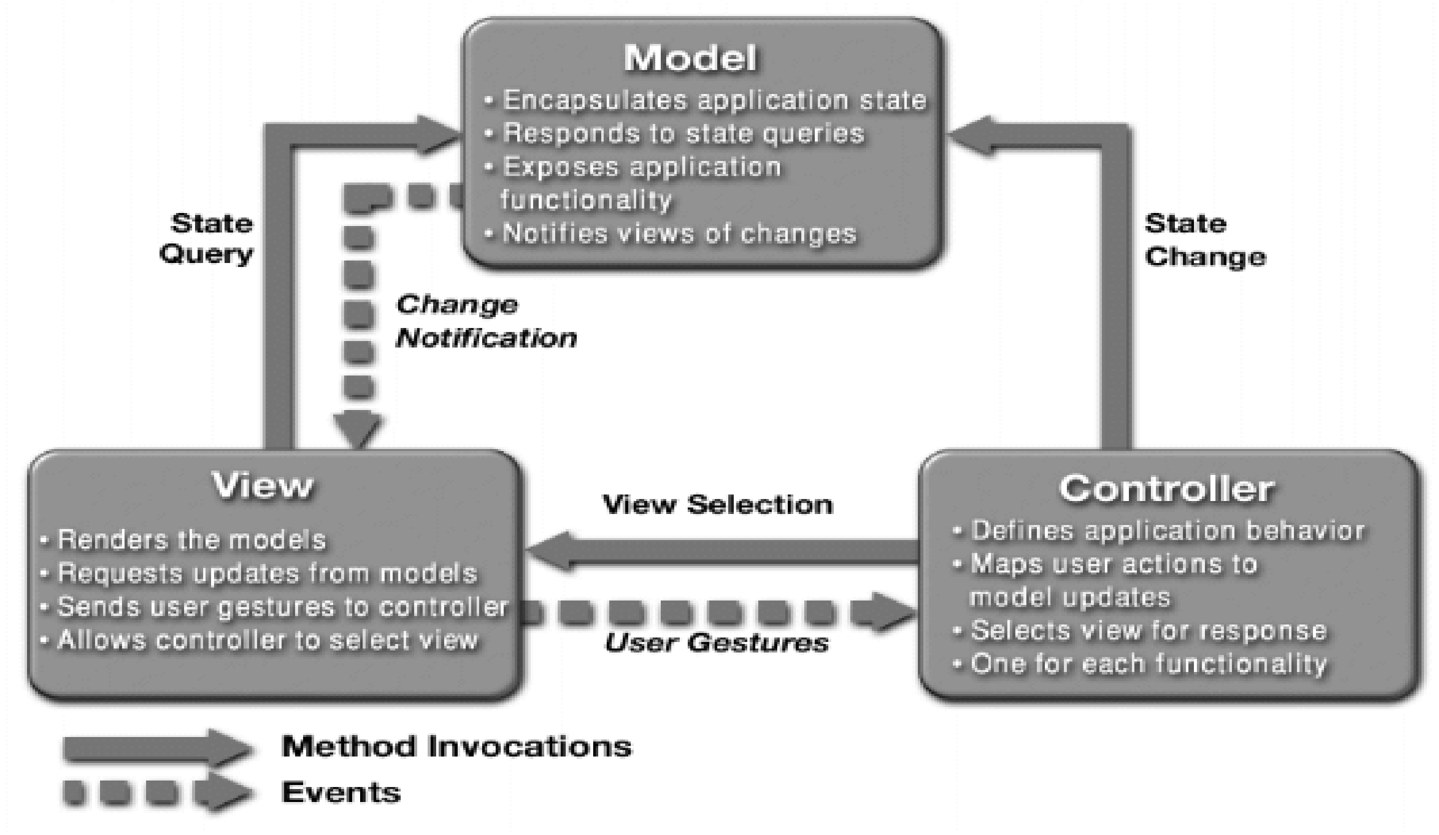
- Saves objects to the database.
- Discovers the columns in a database schema and automatically attaches them to domain objects using metaprogramming.

What is so special about Rails

Action Pack

- Views and controllers have a tight interaction, in rails they are combined in Action Pack
- Action pack breaks a web request into view components and controller compoents
- So an action usually involves a controller request to create, read, update, or delete (CRUD) some part of the model, followed by a view request to render a page

Rails implements the model-view-controller (MVC) architecture.


Model
• Encapsulates application state
• Responds to state queries
• Exposes application functionality
• Notifies views of changes
State Query
Change Notification
State Change
View
• Renders the models
• Requests updates from models
• Sends user gestures to controller
• Allows controller to select view
View Selection
User Gestures
Controller
• Defines application behavior
• Maps user actions to model updates
• Selects view for response
• One for each functionality
Method Invocations
Events

Model View Contoller (MVC)

The MVC design pattern separates the component parts of an application

MVC pattern allows rapid change and evolution of the user interface and controller separate from the data model

Model

- Contains the data of the application
  - Transient
  - Stored (eg Database)

- Enforces "business" rules of the application
  - Attributes
  - Work flow

View

- Provides the user interface
- Dynamic content rendered through templates
- Three major types
  - Ruby code in erb (embedded ruby) templates
  - xml.builder templates
  - rjs templates (for javascript, and thus ajax)

Controller

- Perform the bulk of the heavy lifting
- Handles web requests
- Maintains session state
- Performs caching
- Manages helper modules

Creating a Simple Application

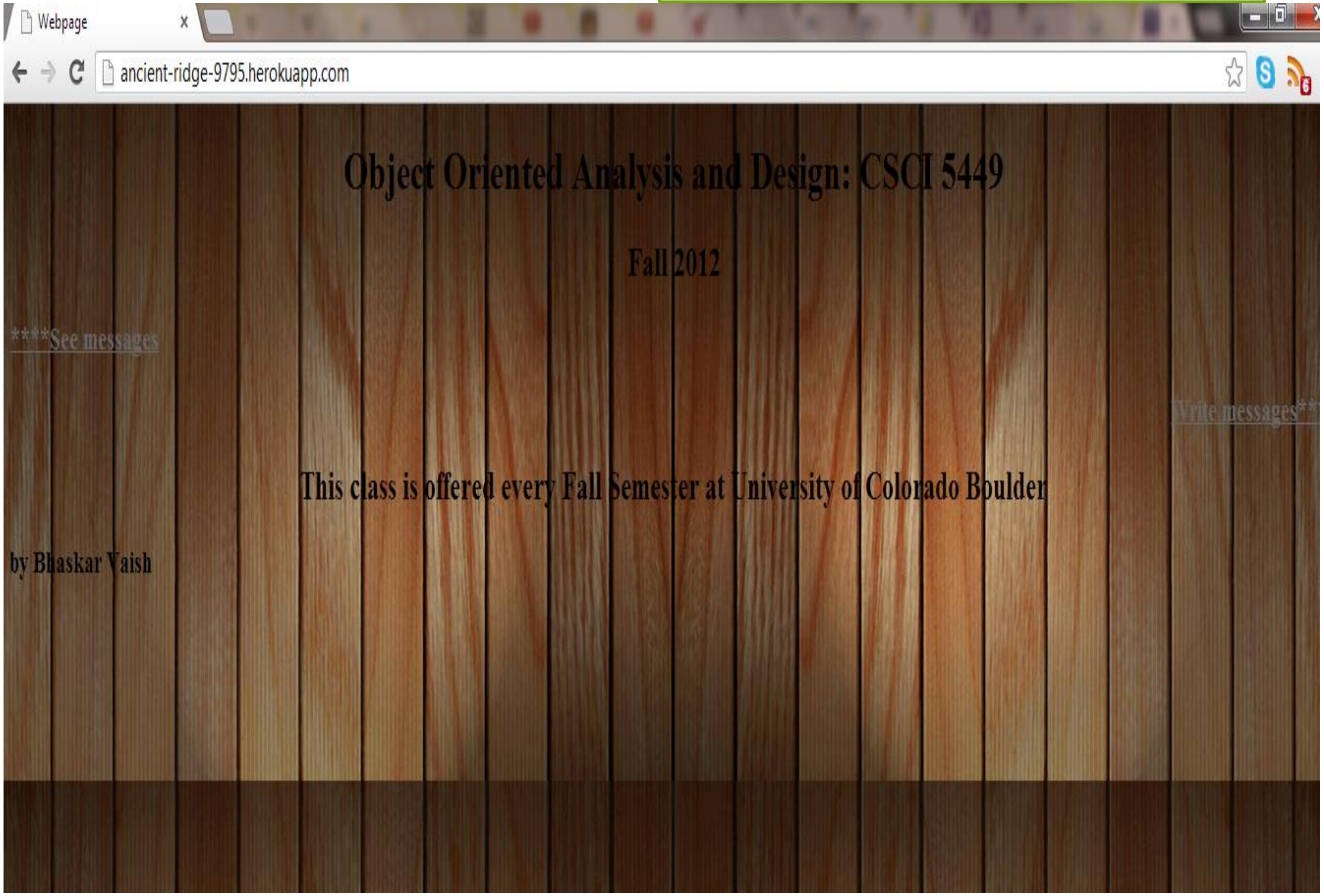
Requirements

- Ruby
- RubyGems
- Rails
- SQlite, PostGres or MySQL

- I will be developing on Windows 7

I have created an application which can be accessed at http://ancient-ridge-9795.herokuapp.com/

It consists of a Home page which displays text And has two right and left buttons, which can be used for displaying and adding messages.



Object Oriented Analysis and Design: CSCI 5449

Fall 2012

****See messages

Write messages**

This class is offered every Fall Semester at University of Colorado Boulder

by Bhaskar Vaish

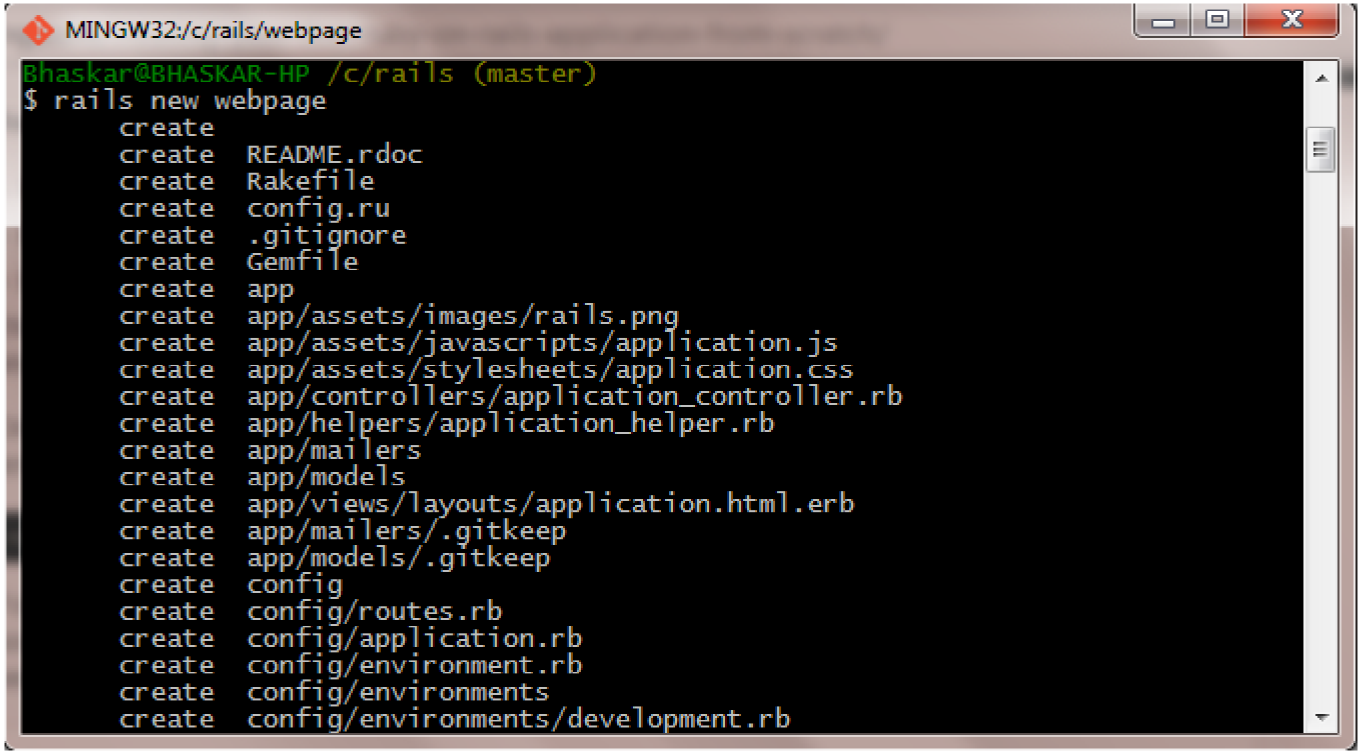
Steps Involved

Creating application on the local host

Step 1: On the Terminal type:

rails new yourapp_name #webpage in my case

hit enter, we see the scripts flow in the terminal creating a bunch of files

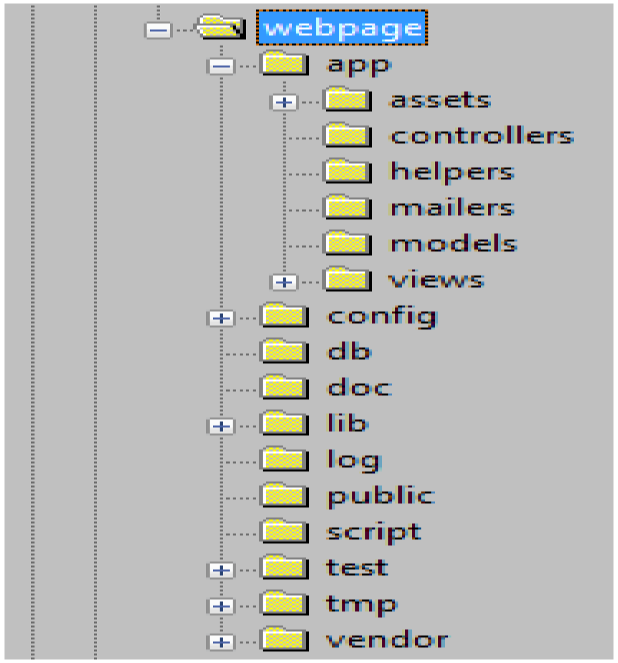


MINGW32:/c/rails/webpage

```
Bhaskar@BHASKAR-HP /c/rails (master)
$ rails new webpage
      create
      create  README.rdoc
      create  Rakefile
      create  config.ru
      create  .gitignore
      create  Gemfile
      create  app
      create  app/assets/images/rails.png
      create  app/assets/javascripts/application.js
      create  app/assets/stylesheets/application.css
      create  app/controllers/application_controller.rb
      create  app/helpers/application_helper.rb
      create  app/mailers
      create  app/models
      create  app/views/layouts/application.html.erb
      create  app/mailers/.gitkeep
      create  app/models/.gitkeep
      create  config
      create  config/routes.rb
      create  config/application.rb
      create  config/environment.rb
      create  config/environments
      create  config/environments/development.rb
```

Step 2: Change the directory to the application

cd webpage



```
webpage
    app
        assets
        controllers
        helpers
        mailers
        models
        views
    config
    db
    doc
    lib
    log
    public
    script
    test
    tmp
    vendor
```

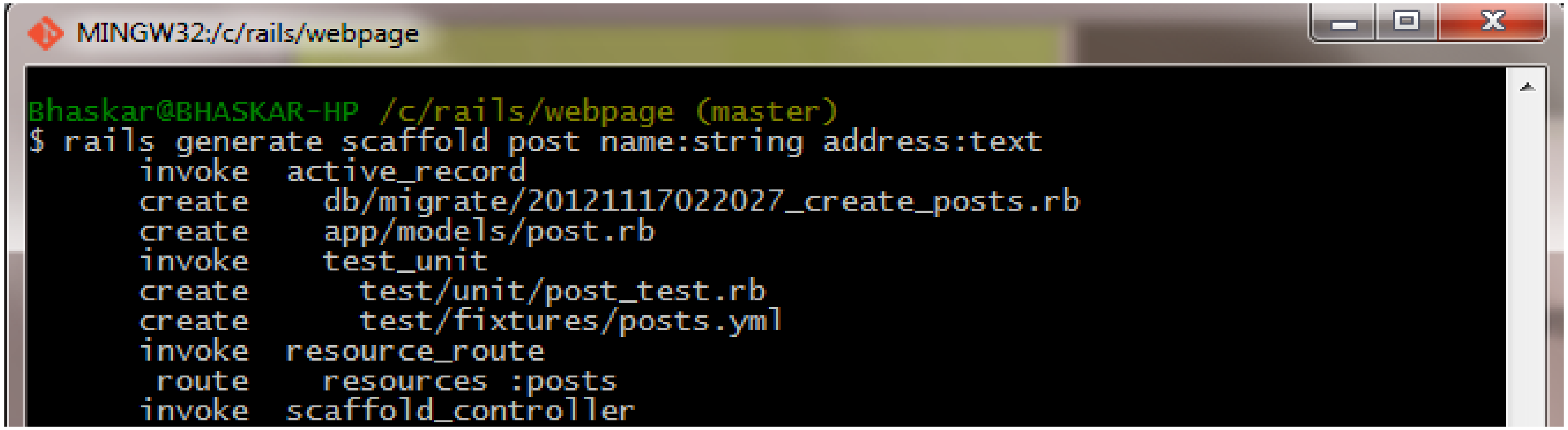
These are the files which rails automatically generates to create the framework for our application.

Step 3:

Create the needed controller, model and views for our application

I will keep simple functionality in which a user can post message

$rails generate scaffold post name:string address:text



MINGW32:/c/rails/webpage

```
Bhaskar@BHASKAR-HP /c/rails/webpage (master)
$ rails generate scaffold post name:string address:text
      invoke  active_record
      create    db/migrate/20121117022027_create_posts.rb
      create    app/models/post.rb
      invoke    test_unit
      create      test/unit/post_test.rb
      create      test/fixtures/posts.yml
      invoke  resource_route
       route    resources :posts
      invoke  scaffold_controller
```

- Scaffold command creates a CRUD (Create, Read, Update, Delete) interface for app (a quick way of creating the MVC automatically).

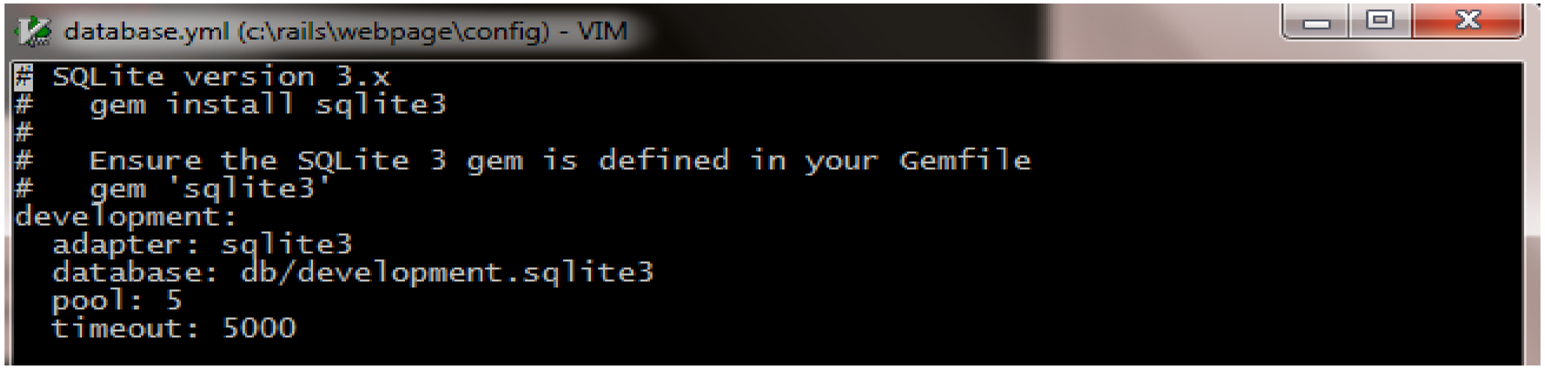
- Alternatively, we can also create our controller, model and view manually using the command

```
// for creating controller
rails generate controller <controller name>
// for creating model
rails generate model <model name>
```

Step 4:

Create Database

rake db:create



```
# SQLite version 3.x
#   gem install sqlite3
#
#   Ensure the SQLite 3 gem is defined in your Gemfile
#   gem 'sqlite3'
development:
  adapter: sqlite3
  database: db/development.sqlite3
  pool: 5
  timeout: 5000
```

The figure shows the database.yml file created

Step 5:

Since we have recently created a new model for Post, a table must be created in our database and requires that we upgrade the database using this command:

rake db:migrate

Step 5:

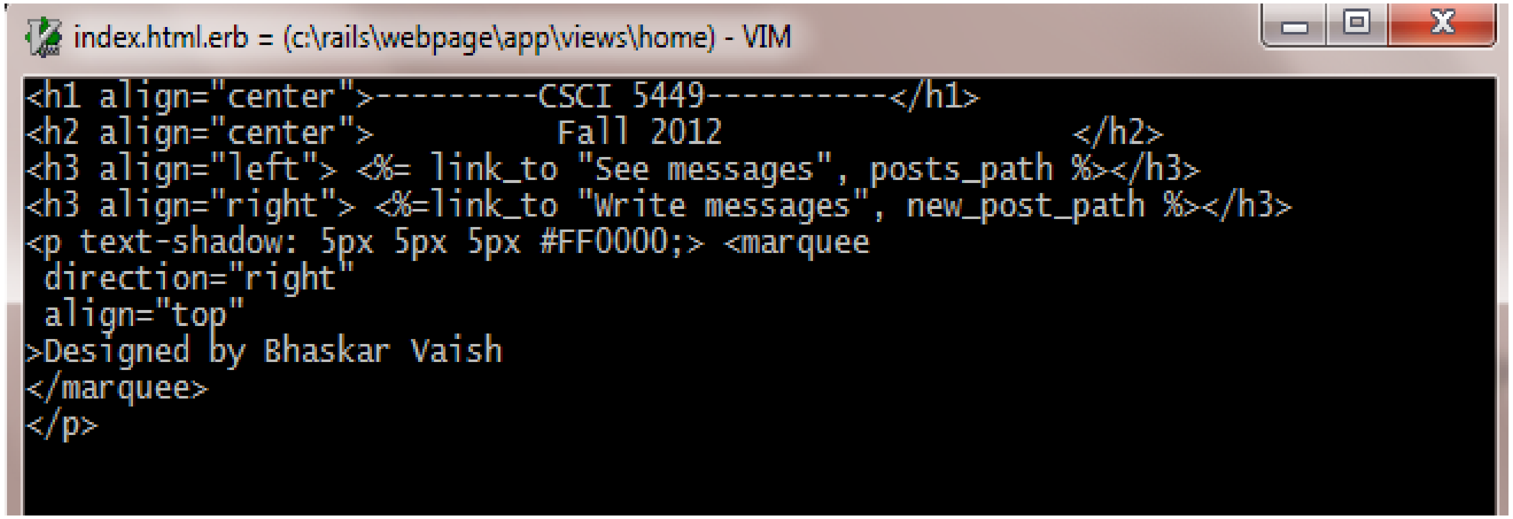
Creating a home page

```
$rails generate controller home index
```

This creates a controller "home" along with views in the app/view/home directory. Rails will create several files for you, including app/views/home/index.html.erb file. This is the template that we will use to display the results of the index action (method) in the home controller. Open this file in your text editor and edit it to contain the code that you want to display in your index page

Editing homepage

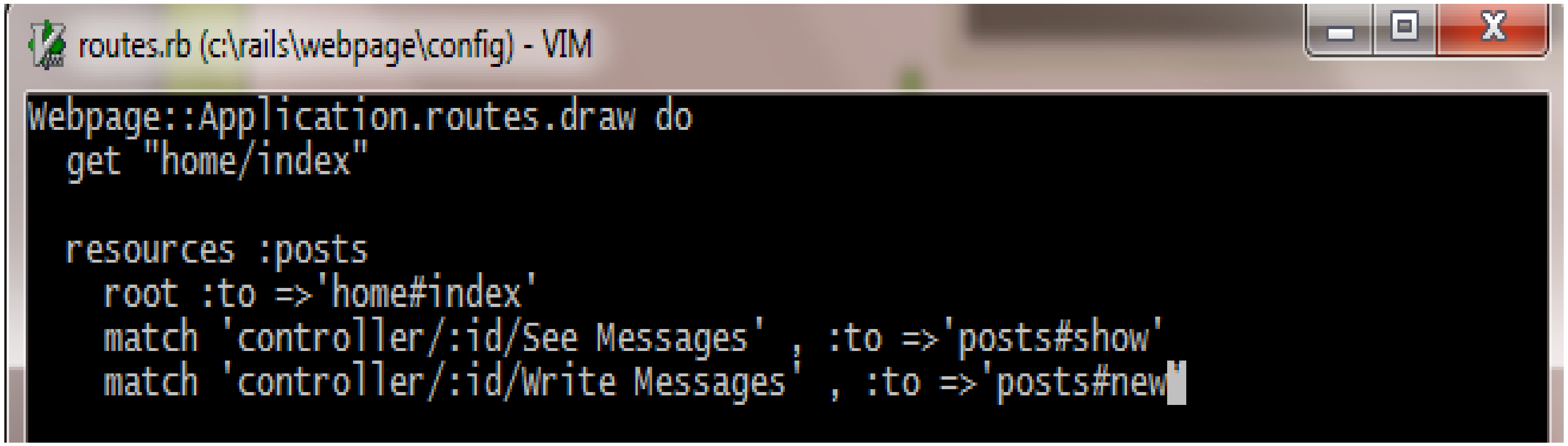
Open file app/views/home/index.html.erb and edit it to contain the code that you want to display in your index page


index.html.erb = (c:\rails\webpage\app\views\home) - VIM

```
<h1 align="center">---------CSCI 5449----------</h1>
<h2 align="center">          Fall 2012                   </h2>
<h3 align="left"> <%= link_to "See messages", posts_path %></h3>
<h3 align="right"> <%=link_to "Write messages", new_post_path %></h3>
<p text-shadow: 5px 5px 5px #FF0000;> <marquee
 direction="right"
 align="top"
>Designed by Bhaskar Vaish
</marquee>
</p>
```

Step 6: Rails Routing

Edit config/routes.rb



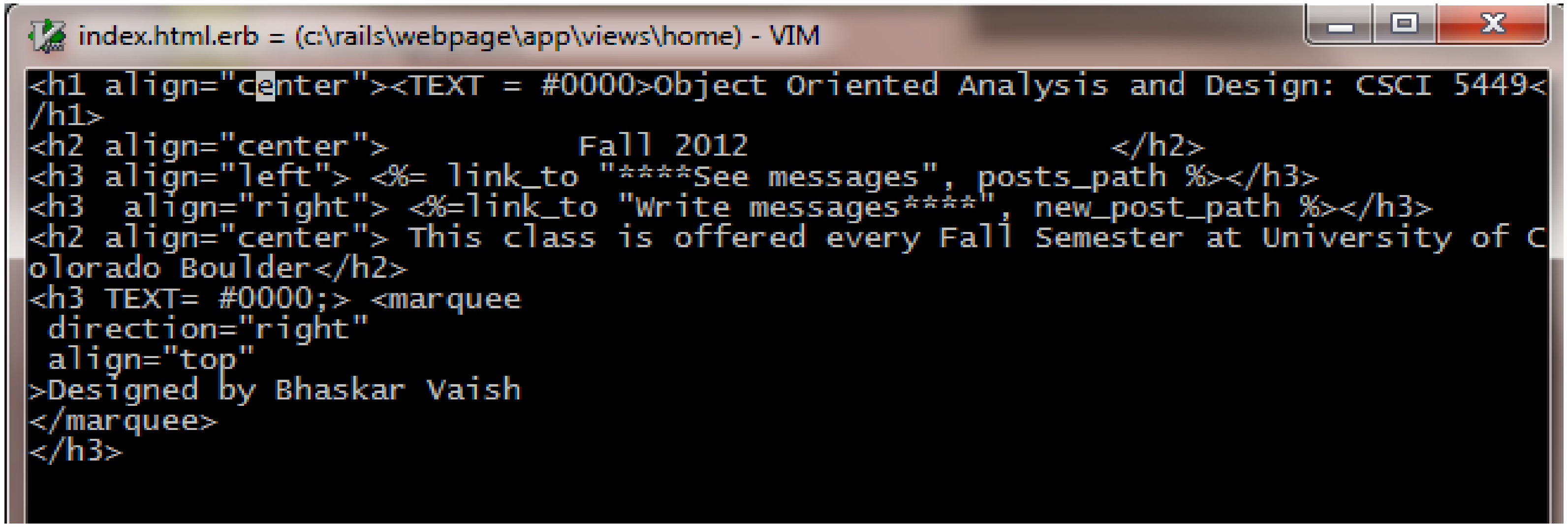
```
Webpage::Application.routes.draw do
  get "home/index"

  resources :posts
    root :to =>'home#index'
    match 'controller/:id/See Messages' , :to =>'posts#show'
    match 'controller/:id/Write Messages' , :to =>'posts#new'
```

The See Messages and Write Messages will appear on the main page

Step 7: Text on the page

Edit the index.html file, enter text which will appear on the screen

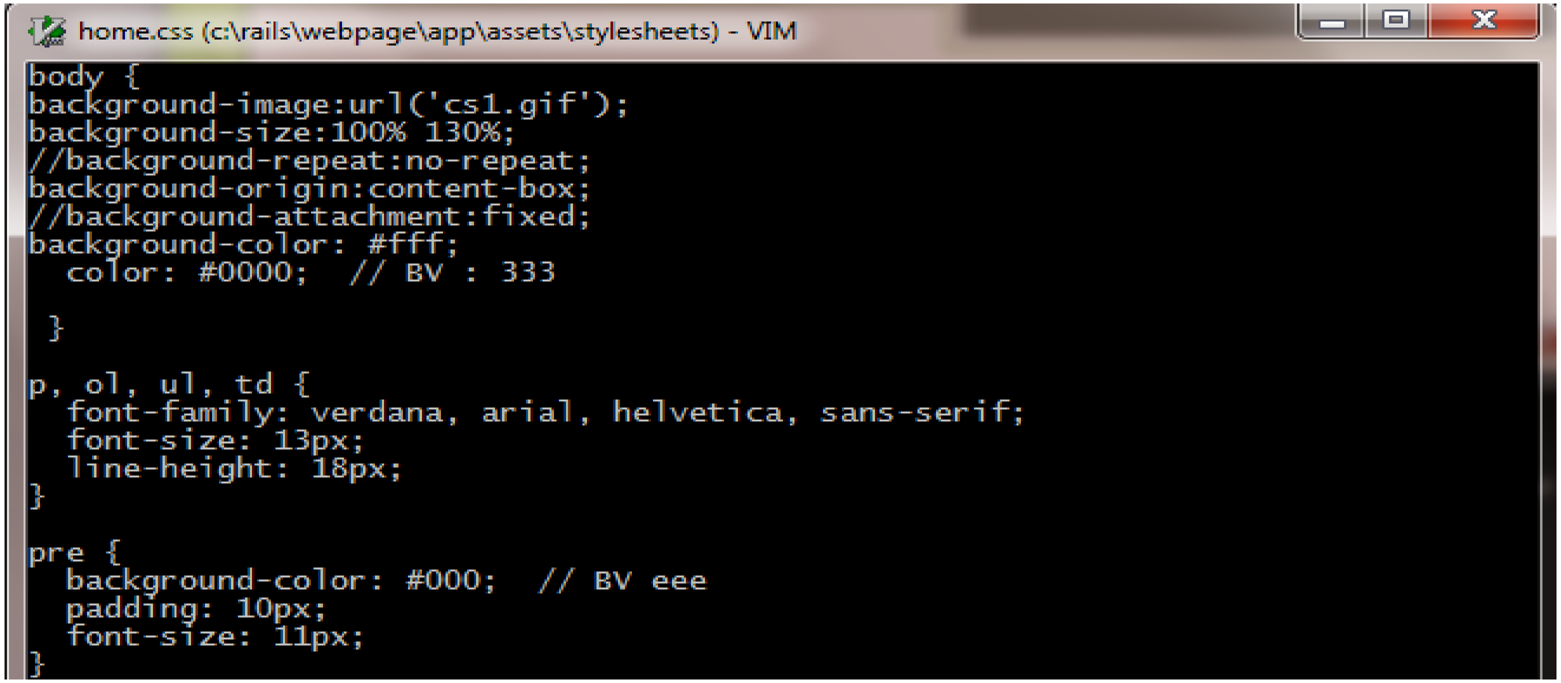


```
<h1 align="center"><TEXT = #0000>Object Oriented Analysis and Design: CSCI 5449<
/h1>
<h2 align="center">          Fall 2012                   </h2>
<h3 align="left"> <%= link_to "****See messages", posts_path %></h3>
<h3  align="right"> <%=link_to "Write messages****", new_post_path %></h3>
<h2 align="center"> This class is offered every Fall Semester at University of C
olorado Boulder</h2>
<h3 TEXT= #0000;> <marquee
 direction="right"
 align="top"
>Designed by Bhaskar Vaish
</marquee>
</h3>
```

The See Messages and Write Messages will appear on the main page

Step 8: Set Background images and color

Edit the home.css file, to set background image and text color


home.css (c:\rails\webpage\app\assets\stylesheets) - VIM

```
body {
background-image:url('cs1.gif');
background-size:100% 130%;
//background-repeat:no-repeat;
background-origin:content-box;
//background-attachment:fixed;
background-color: #fff;
  color: #0000;  // BV : 333

 }

p, ol, ul, td {
  font-family: verdana, arial, helvetica, sans-serif;
  font-size: 13px;
  line-height: 18px;
}

pre {
  background-color: #000;  // BV eee
  padding: 10px;
  font-size: 11px;
}
```

The See Messages and Write Messages will appear on the main page

Step 8: Testing the application

On the command line enter

```
rails server
```

The application will be uploaded to http://localhot:3000/

References:

- ppt, Ruby on Rails, A new gem in web development
- ppt, Ruby Intro
- Ruby on Rails tutorial book

http://en.wikipedia.org/wiki/Ruby_on_Rails

http://fuelyourcoding.com/creating-your-first-ruby-on-rails-application-from-scratch/