Project Report: User Guide Web application

By: Liyi Zhang Chu Zhang

I Project Abstract

The theme of the designed project is to provide tour guides with searching function about tourist destinations around different areas of America, and sharing travel notes and comments among users.

The guide about one destination contains brief introduction (profiles, locations, best visiting time, wearing suggestions, telecommunication, etc) and also all the scenery spots about one destination. There are also travel notes written by users about one destination. The users can write comment on each scene or travel note, as well as write travel notes about a destination, together with the uploading photos function. Users (login or not) can search destinations/scenes/travel notes by any key words.

Main pages are as follows: Home page displaying all the destinations (with their location and profile); Destination page displaying brief introductions about one destination----scenes tab displaying all the scenes (with profiles and comment numbers) about the destination, travel notes tab displaying all the travel notes about one destination (with sorting function: by most reviewed or by latest posted); Scene page displaying a specific scene with all the comments: Travel note Page displaying one specific travel note with its contents and all the comments (users can write reviews and upload pictures on this page); SearchResult Page displaying all the results (destiations/scenes/travel notes) containing the key words.

The project uses Java programming language and MySQL database, uses Eclipse as the development environment, uses Maven to build the project, and also utilizes Spring Framework to ease the design. We especially use the Spring MVC framework to utilize MVC design pattern, and other Spring frameworks like Spring JDBC Template.

II Implementation

1. First, users should download Apache Maven 3.3.3 from the official website <https://maven.apache.org/download.cgi>. Then install and deploy Maven as following steps:

* Start up Terminal
* Type “cd ~/” to go to home folder
* Type “touch .bash\_profile” to create new file
* Edit .bash\_profile with editor (or type “open –e .bash\_profile” to open it in TextEdit)

export M2\_Home=usr…../apach-maven-3.33 (the location of maven)

export PATH=$PATH:$M2\_Home/bin

* Type “. .bash\_profile” to reload .bash\_profile and update any function added

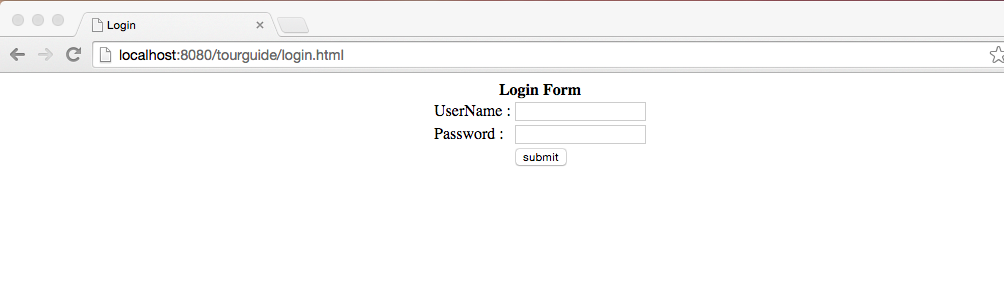
1. Enter the project directory on the terminal, and type the commands as follows, to start the web application

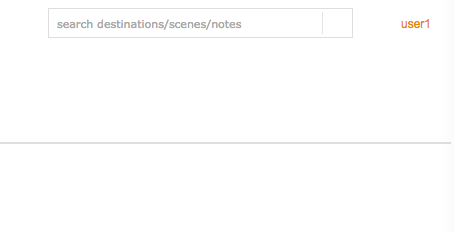
$ mvn clean install

$ mvn tomcat7:run.

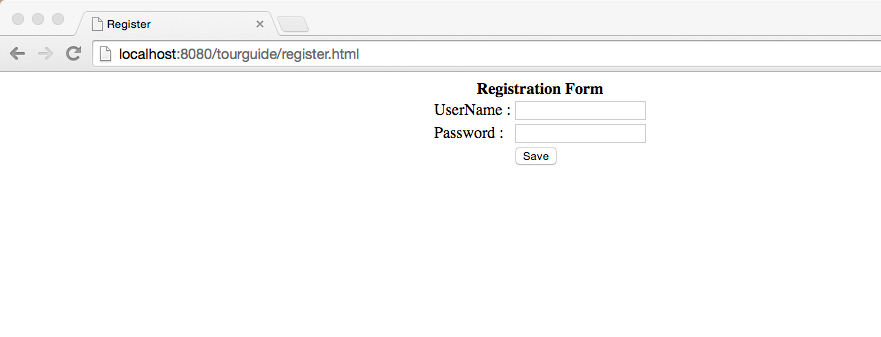
3. The web application running results are as follows:

Login page

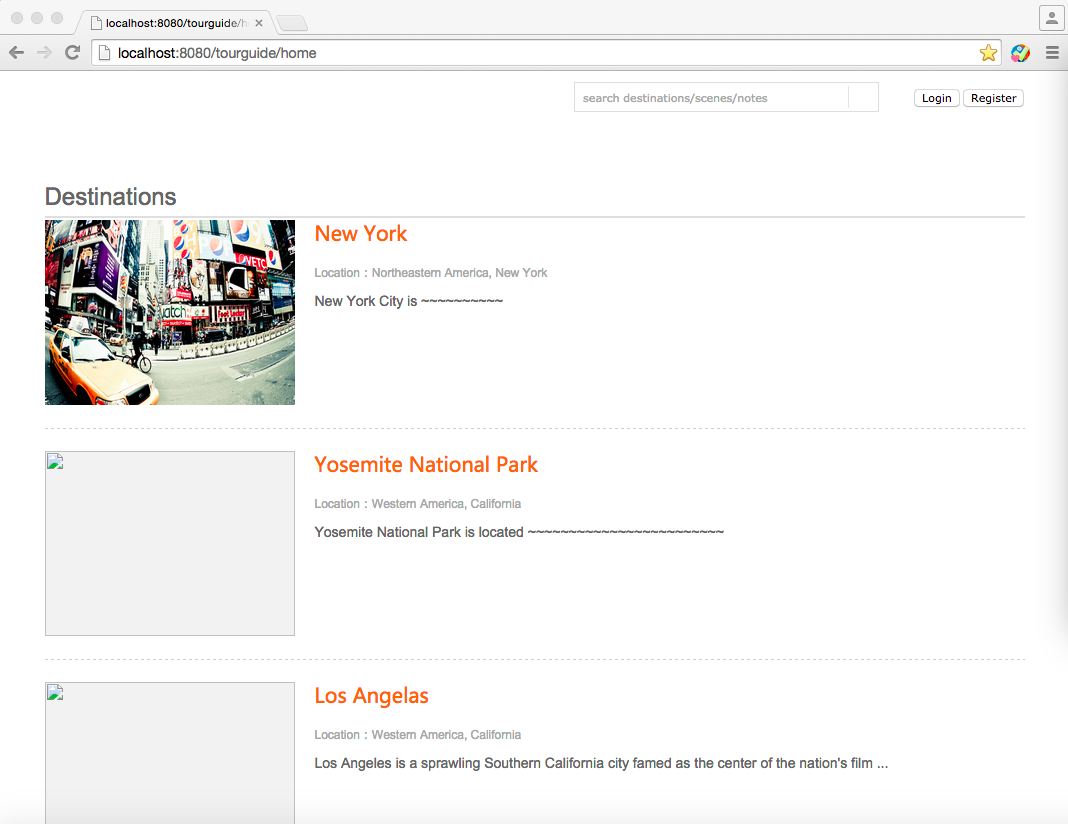


once logged in, the session will be set up and logged-in state will be kept.

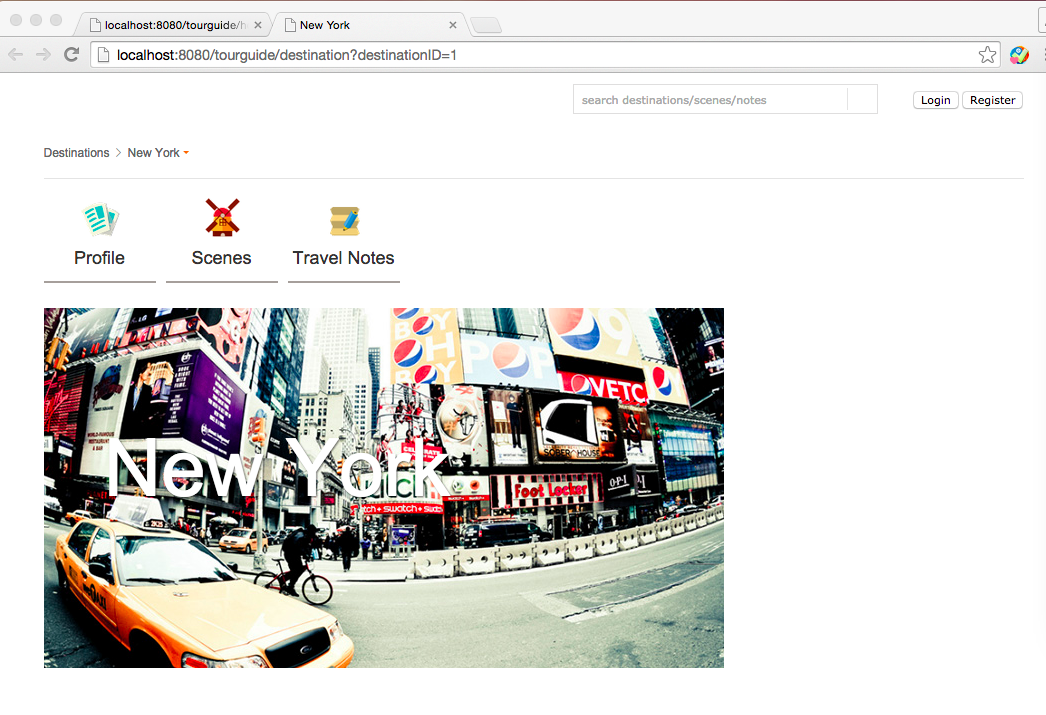
Registraion Page

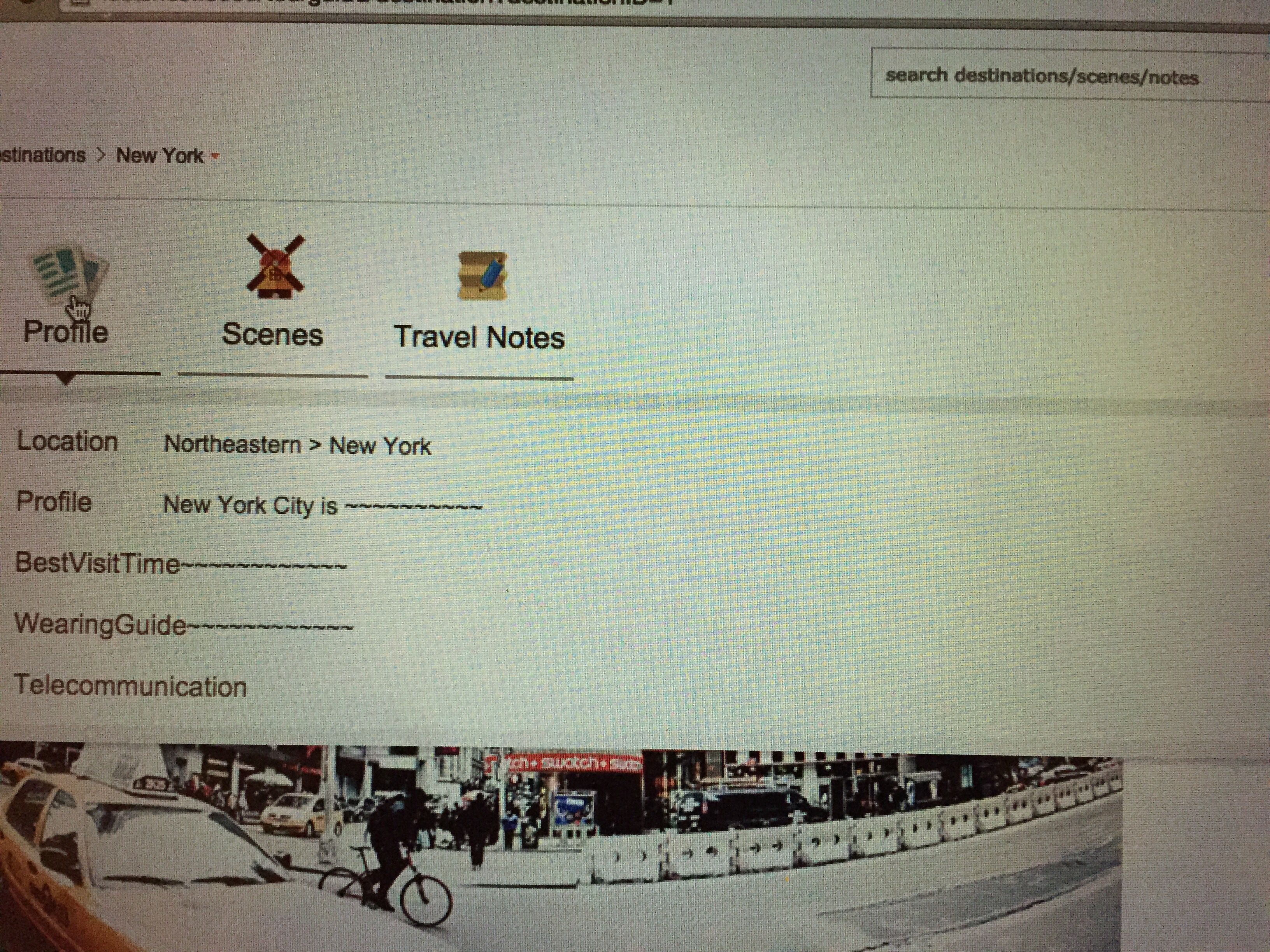


Home page



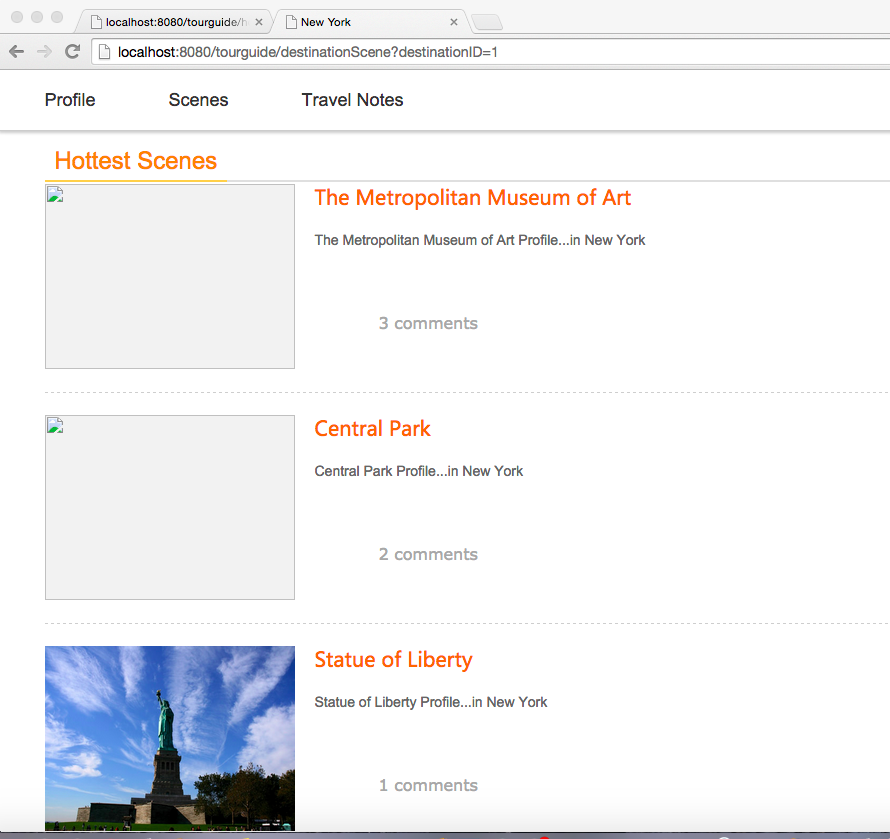
Destination page



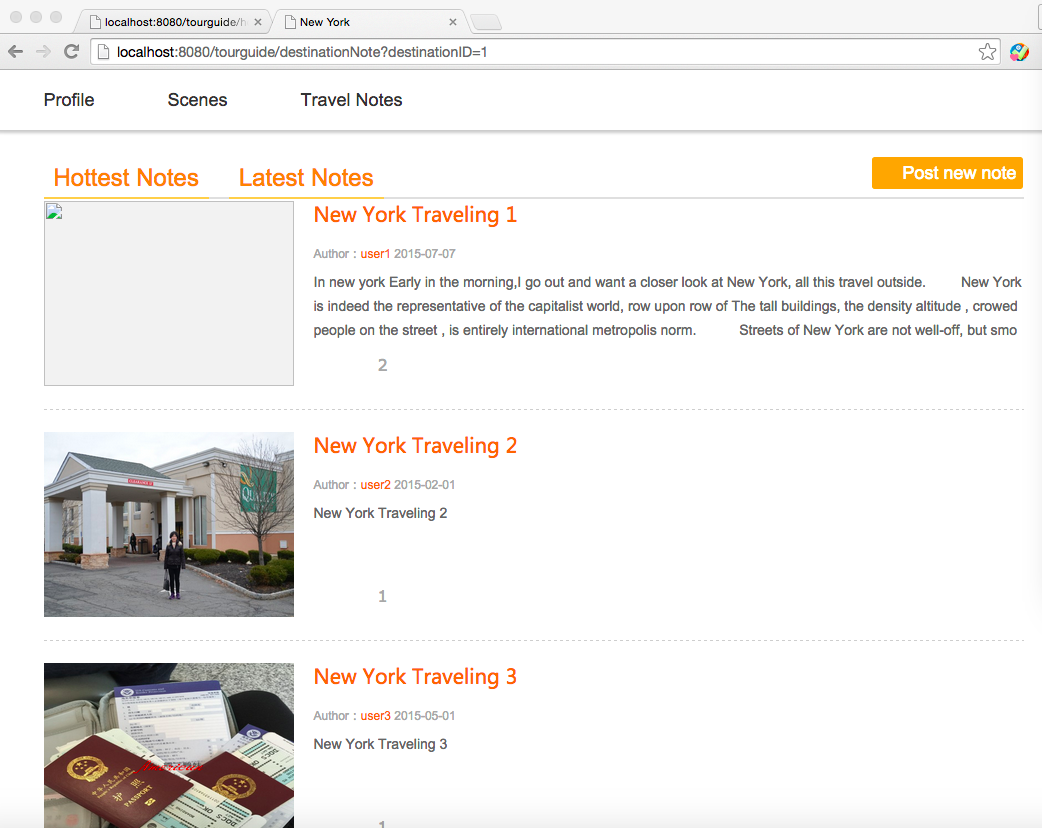


mouse pointer on profile will display the profile in drop-down.

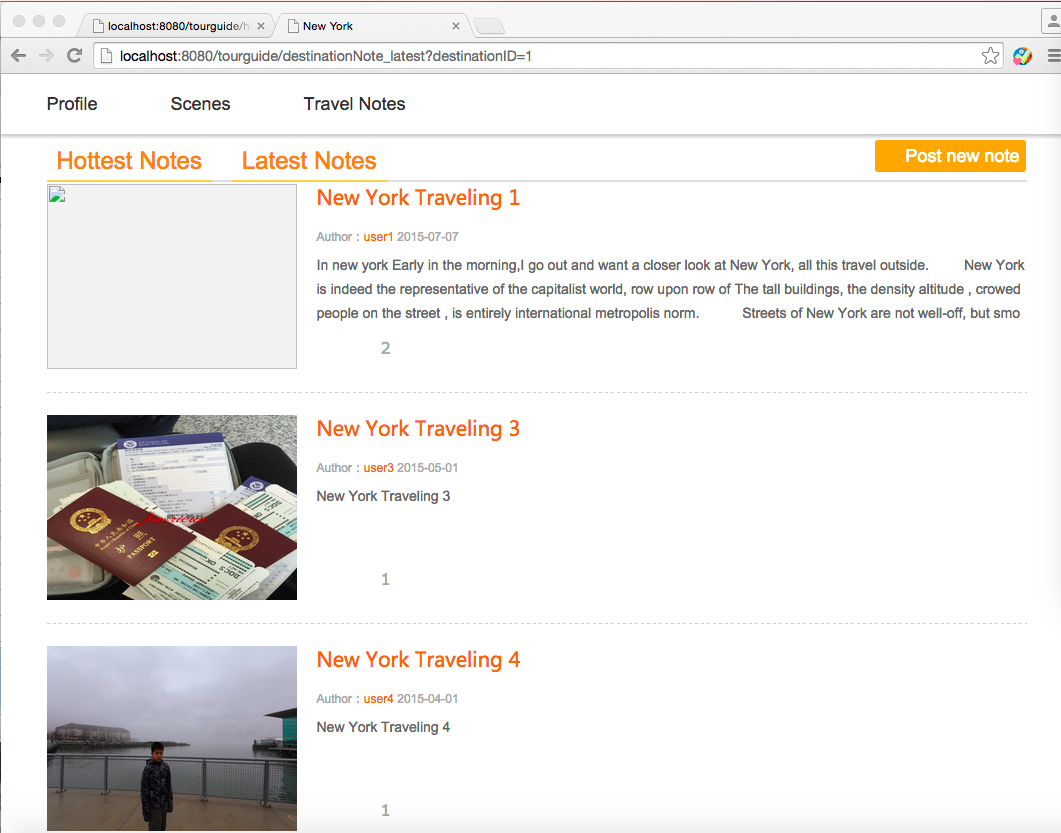
Destination SceneList Tab:



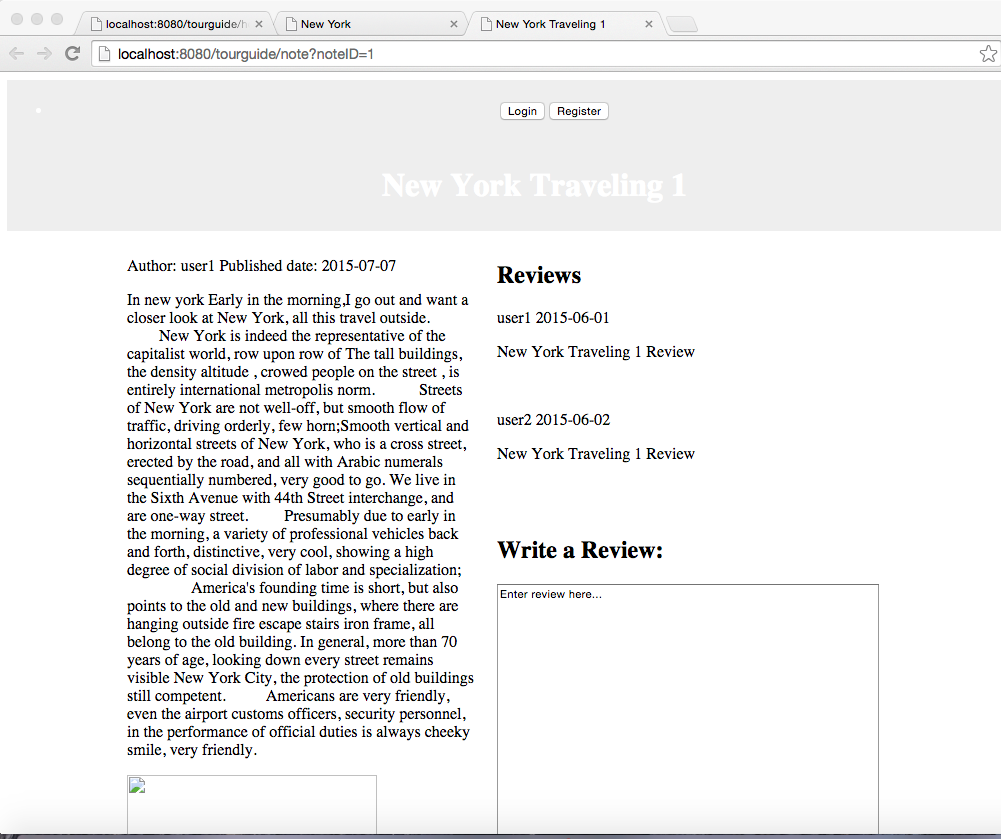
Destination TravelNoteList Tab: (sorted by comment numbers by default)

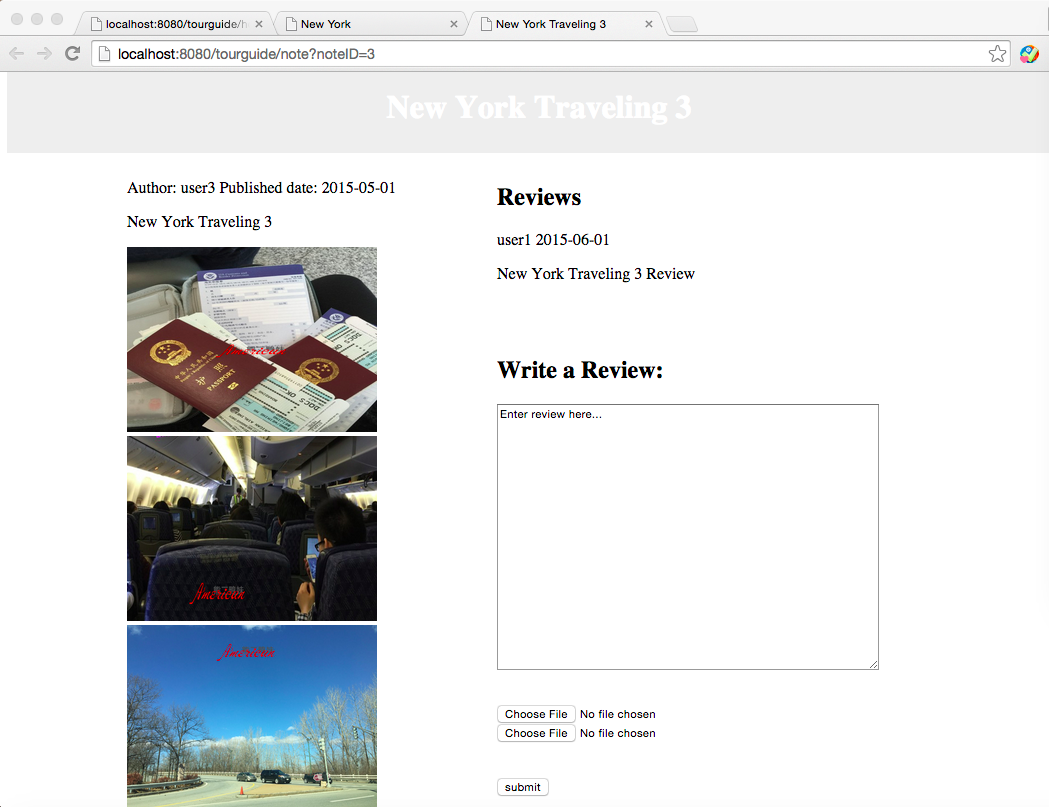


Destination TravelNoteList Tab: (sorted by posting date)



TravelNote Page:

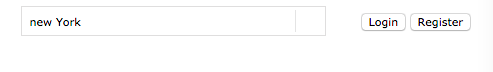


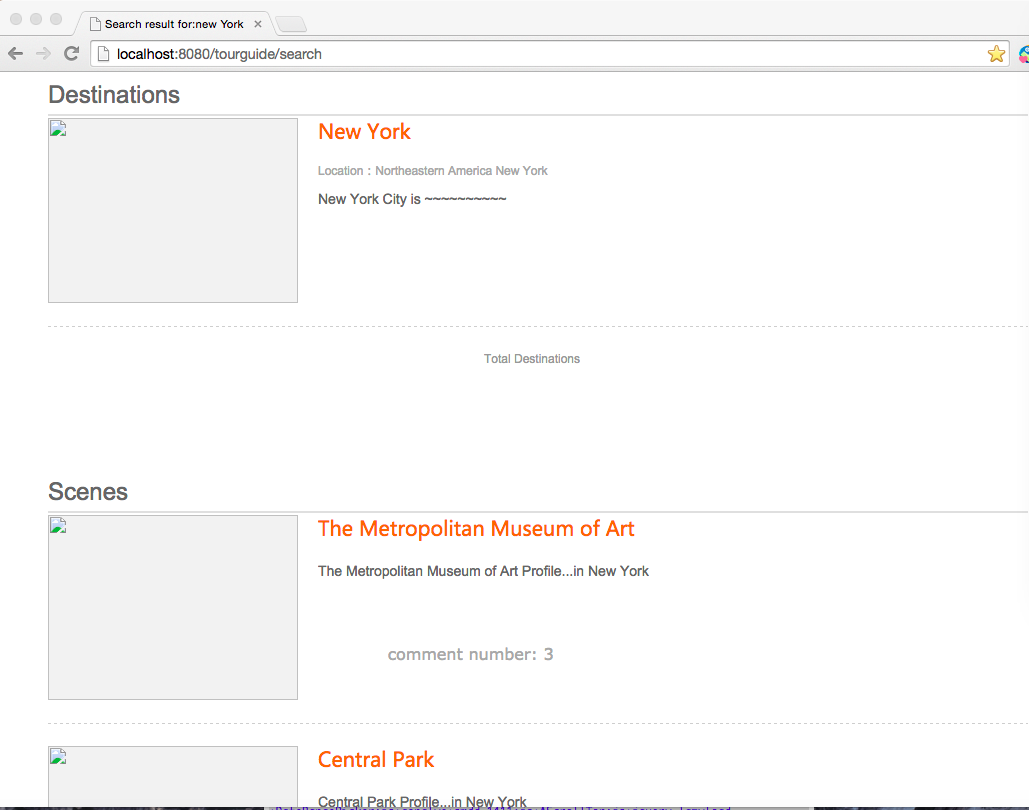


Once reviewed, the comment will display among the comments:

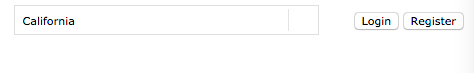


Search for: New York





Search for: California





III project framework and structure

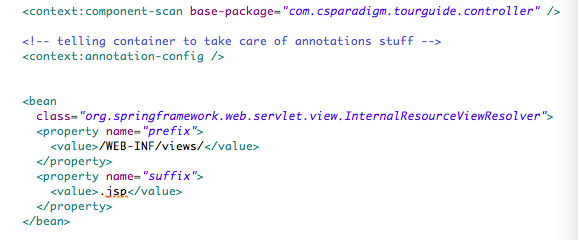
A Spring MVC Web Application using Spring JDBC template This web application is developed using Java language on Eclipse IDE and utilizes the Spring MVC framework with maven’s project object model (POM.xml) and Spring JDBC framework with MySQL relational database. The Maven is used to build and install the project and the maven-tomcat-plugin is used to run the installed .war file. Also, it greatly simplifies the implementation of Spring MVC design pattern.

**Deployment Descriptor: web.xml**

It defines all the Spring Bean Configuration Files in our case the **servlet-context.xml and root-context.xml**. Spring framework is plugined into our web application by configuring Spring framework DispatcherServlet as front controller, which is the spring container shared by all Servlets and Filters, who process the application requests.



**Servlet-context.xml:**

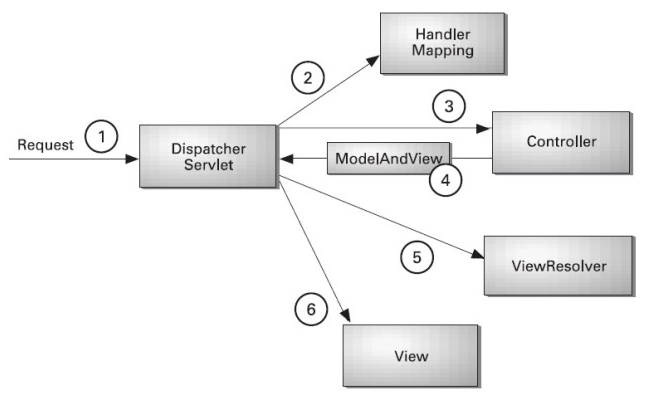
****

****

Front Controller: base-package specifies the package of the controllers. prefix specifies the directory of views, and it is set to be /WEB-INF/views/. suffix specifies the file extension of views. So it Resolves views selected for rendering by @Controllers to .jsp resources in the /WEB-INF/views directory

So, after receiving an HTTP request, DispatcherServlet consults the HandlerMapping to call the appropriate Controller. The Controller takes the request and calls the appropriate service methods. The service method will set model data based on defined business logic and returns view name to the DispatcherServlet.

The DispatcherServlet will take help from ViewResolver to pickup the defined view for the request. Once view is finalized, the DispatcherServlet passes the model data to the view which is finally rendered on the browser.



Also defines datasource for JDBC template.

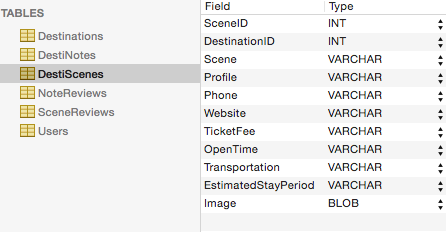
**View Pages**

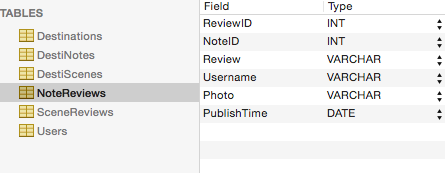
All the jsp pages.

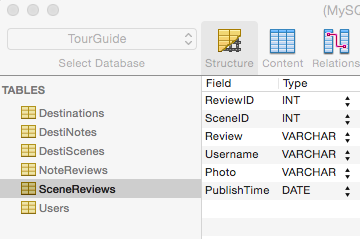
**MySQL Tables:**

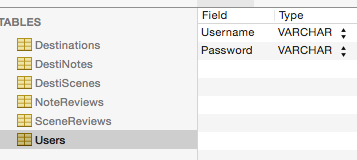
****

****

****

****

****

****

**All the java classes:**

MainController.java is Controller class, which handles the requests (e.g. *@RequestMapping*) and return the view pages. The attributes added to Model in Controller handler methods can be used to create view pages. JSP page uses Spring form tags to bind the values of the form to a model object(@ModelAtrribute). @SessionAttributes("userObj") can share object among all model and views, not just in one mapping.

User.java, Destination.java, Scene.java, Note.java define the Model or Entity Bean. This class simply maps a row in the database table selecting result to a plain old Java object (POJO) -User.

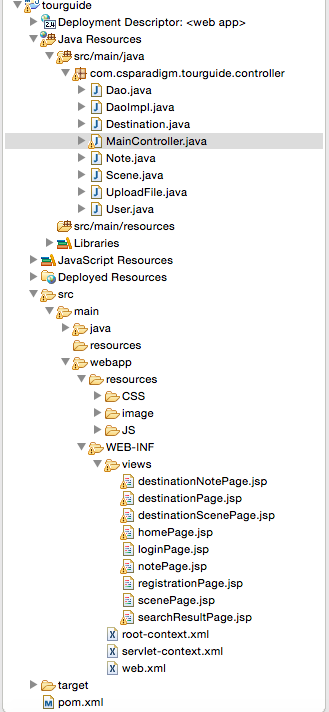
Dao.java is UserDao interface, which defines methods for performing CRUD operations on the database tables.

DaoImpl.java is DAO (Data Access Object) Implementation class ofDao interface.

DaoImpl.java here represents actual coding to deal with jdbc template. The autowired datasource bean here is passed to get an object of JdbcTemplate class on. Then call appropriate methods on this JdbcTemplate object to manuplate data in DB. jdbcTemplate.query(sql, new Mapper()) method will use the String SQL to query the database and will loop through each row in the ResultSet. UserMapper implements RowMapper to map a row in the ResultSet to a POJO object, and UserExtractor implements ResultSetExtractor to extract a single row as a POJO.

**Resources:**

All the pictures, css files, js files are retrieved using relative path, image address is stored and retrieved into database.



Reference：

<http://o7planning.org/web/fe/default/en/document/8108/spring-mvc-for-beginners>

<http://www.beingjavaguys.com/2013/07/spring-jdbc-template-with-spring-mvc.html>

<http://jeromejaglale.com/doc/spring4_tutorial/display_process_form>

<http://www.tutorialspoint.com/spring/spring_web_mvc_framework.htm>