Design Document

My Movie Theater

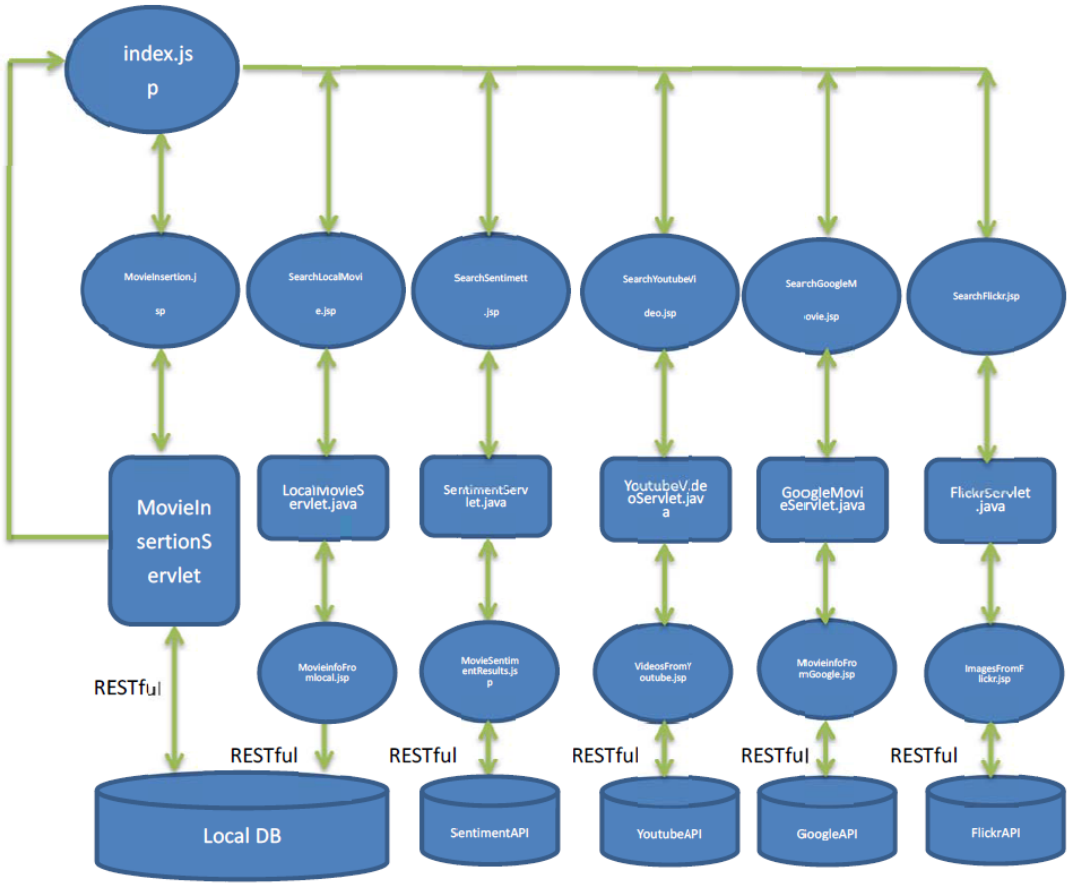
2015/8/15

Yu Jun 26346702

Design Document for My Hotel Booking System

Content

1. The Model of the Movie Theater



Comments: You can see from the above architecture that I just have implemented five API: one is local DB which is achieved by using RESTful service, and the others are from Viraheat.com, Youtube.com, Google.com and Flickr.com. All of them except the local DB require you to apply an API key from these Internet Companies. The only thing you need to do is using their provided API to implement the requirements for this assignment. The JSP files are used to display the Web UI for the users to interact with them. When the user submits their results to the JSP files, the servlets are triggered to pass forward the arguments required to get the results from local DB or the Internet – Based API to the resulting JSP files to display the results in a way the user feel more comfortable to the interacting users.

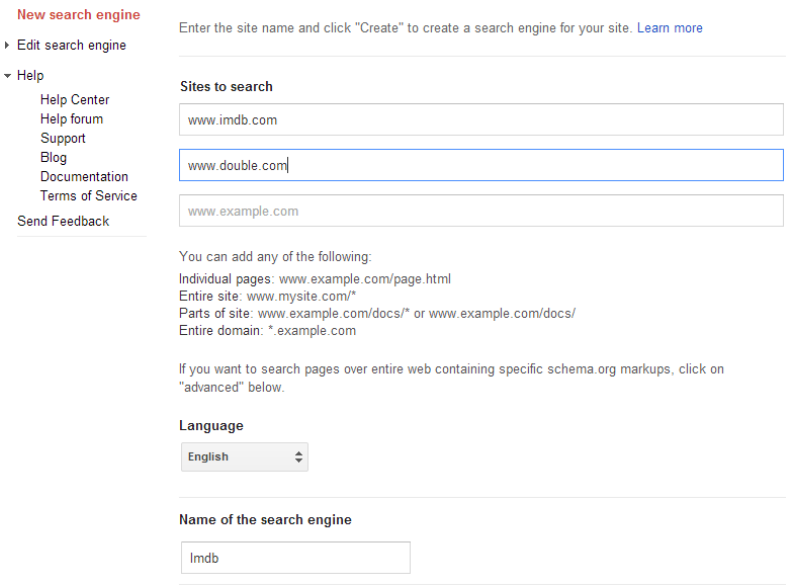
2. The Tasks I have accomplished:

1) The users are able to search the movie information from the local database through RESTful services generated from database entities. In my own Database, I have provided ten movies information for you to search. While you could also add your movie information to my website whenever you want. This is function I would talk about later

in this document. You should fill in the blank with movie name you want to search, if the movie exists, you would be able to see the relevant information from my website. I have changed the MovieFacadeREST.java a little bit to enable the users to search the movie information based on the movie name as you can from the following figure:

```
@GET
@Path("{movieName}")
@Produces({"application/xml","application/json"})
public Movie find(@PathParam("movieName") String movieName)
    int index = -1;
    Movie movieInstance = new Movie();
    List movielist = null;
```

2) Similarly, you are able to search the movie information through the general search using the google customer search within the www.imdb.com and www.douban.com, which means that you would only see the information from these two websites using the google search. Or put it in another way the google customer search would only display the movie information from these two websites. The procedure to customize you own search provided by the google is as following figure:



You can fill in the sites to search blank with www.imdb.com and www.douban.com. Thanks to that you would be more likely to get the relevant movie information from my customer search. After you have done that, you would get a “CX” after completion of application. The “CX” is used to retrieve information from the google customer search together with your API key.

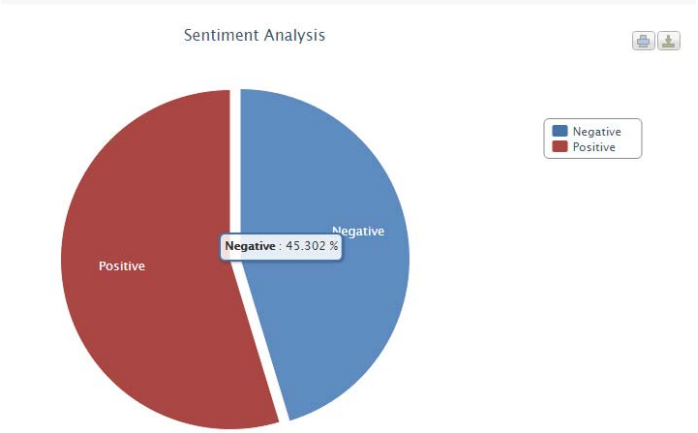
3) The Web- Based Client that I provide enables you to access and enjoy the functionalities in a way you may feel more ease and friendly. You would be able to search photos, movie information from both local and google customer search, videos from youtube.com which is in trail form. What’s more, you would be able to add you favorite movie information to my local web site. Last but not least, you can get the sentiment results of a particular movie from the API provided by the viralheat.com.

4) Using the API provided by the flickr.com, you can search for the photos relevant to the movie name. Like before, you should also apply an API key from flickr.com before you use the functions. In my web site, you would be able to see 12 relevant photos.

5) If you are more interested in a particular movie, you can fill in the blank with the movie name you are interested. You are able to see 8 relevant videos from my web site. What's more interesting and fantasy is that you would only see trails of a particular movie.

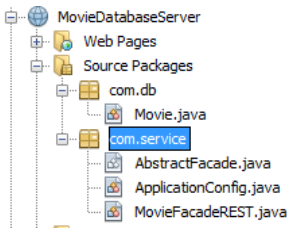
6) The users would be able to add their own movie information into the local database through RESTful service. Movie duplication must be avoided, so the web site would firstly check whether the movie name have already existed in the local DB. If so the web site would return to the homepage, if not you would be able to go to site for you to fill the movie information.

7) Sentiment analysis is enabled in my web site. If you are eager to know the popularity of a movie you are interested, you can fill in the blank with the movie name. Eventually, you would see both positive and negative percentage of a specific movie in a chart form like the following:



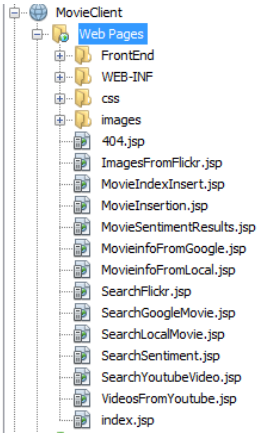
You would be able to download this chart from my web site with four formats by clicking upper - right corner button.

3 The main Class of the movie program:

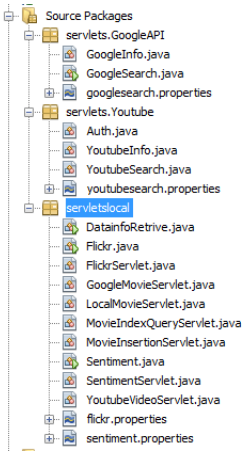


1) In the MovieDatabaseServer project, The Movie.java class is generated through the way of 'Entity classes from Database', while you should establish database

connection first. The classes in the package com.service like before are generated through the way of 'RESTful Web Services from entity classes'. So in this step, you don't have to worry too much about the classes, you only need to care about the configurations. You just need to add a method into the movieFacadeREST.java, which would allow you to search for a particular movie with a specified movie name.



2) In this project, I just have written a Web Client of 15 JSP files allowing the interaction between users and the server. The index.jsp file is the homepage of my web client. 404.jsp is used to display the error information when the server occasionally encounter an unexpected error. In the FrontEnd folder, there are css and JS files to display the contents more comfortable and beautiful.



3) These are the servlets and API web services provided by the internet company, the properties files are containing the API keys for these web services only for the consideration of security issues. Servlets are used in my case to redirect the request and pass forward the arguments to the other JSP files required to display the results of the requests.

4 The configuration files for the Web Services

The JDBC – connection –pool and JDBC – resource are needed for successfully deploying the web service. Therefore, if we want to deploy the war files to another environment or other hosts, we should first provide these configuration requirements as following link:

```
<jdbc-connection-pool datasource-classname="com.mysql.jdbc.jdbc2.optional.MysqlDataSource" name="mysql_id26346702_fit5192a1Pool" wrap-jdbc-objects="
false" connection-validation-method="auto-commit" res-type="javax.sql.DataSource">
  <property name="serverName" value="localhost"></property>
  <property name="portNumber" value="3306"></property>
  <property name="databaseName" value="id26346702"></property>
  <property name="User" value="fit5192a1"></property>
  <property name="Password" value=""></property>
  <property name="URL" value="jdbc:mysql://localhost:3306/id26346702?zeroDateTimeBehavior=convertToNull"></property>
  <property name="driverClass" value="com.mysql.jdbc.Driver"></property>
</jdbc-connection-pool>
<jdbc-resource pool-name="mysql_id26346702_fit5192a1Pool" jndi-name="MovieDB"></jdbc-resource>
```

```
<application-ref ref="MovieDatabaseServer" virtual-servers="server"></application-ref>
<application-ref ref="MovieClient" virtual-servers="server"></application-ref>
```

Note: if you can't see the figure clearly you can open the file named jdbc-pool.txt right in the same folder as this document.

The Local Web service address is as following:
http://localhost:8080/MovieDatabaseServer/test-resbeans.html

The application.wadl for the Local Web Service:
http://localhost:8080/MovieDatabaseServer/webresources/application.wadl

The Web Client address for My Movie Theater:
http://localhost:8080/MovieClient/index.jsp