2015

Design Documentation

ONLINE MOVIES SEARCH APPLICATION

FIT5192 ASSIGNMENT ONE

26346990 BAI JUN

Contents

1. Task Overview

Task 1: I have written a RESTful web service that accepts requests for a certain movie by its title and retrieves relevant information based on the requests from a MySQL database. There are 11 movies (information) that already stored in the database.

Task 2: I have written a web-based client search application that accesses the web service I created in Task 1, together with Google, YouTube and Flickr for the requested DVD/movie item. My GUI have the ability to play the video for the user (I embed the video in my client).

Task 3: I have extended my web service with an extra operation that allows the user to add new movies to the database. This also involves including an option in the web-based client application to add new movies.

Task 4: I have extend my work by performing a simple sentiment analysis using Dandelion Sentiment Analysis API. It can tell whether the expressed opinion in short texts is positive, negative, or neutral. Given a short sentence, it returns a label representing the identified sentiment, along with a numeric score ranging from strongly positive (1.0) to extremely negative (-1.0).

2. Develop Environment

- JDK 1.7;
- MySQL 5.6;
- Glassfish 4.1;

Before deploying this project, you should install and properly configure all the develop environment software above.

3. System Architecture Diagram

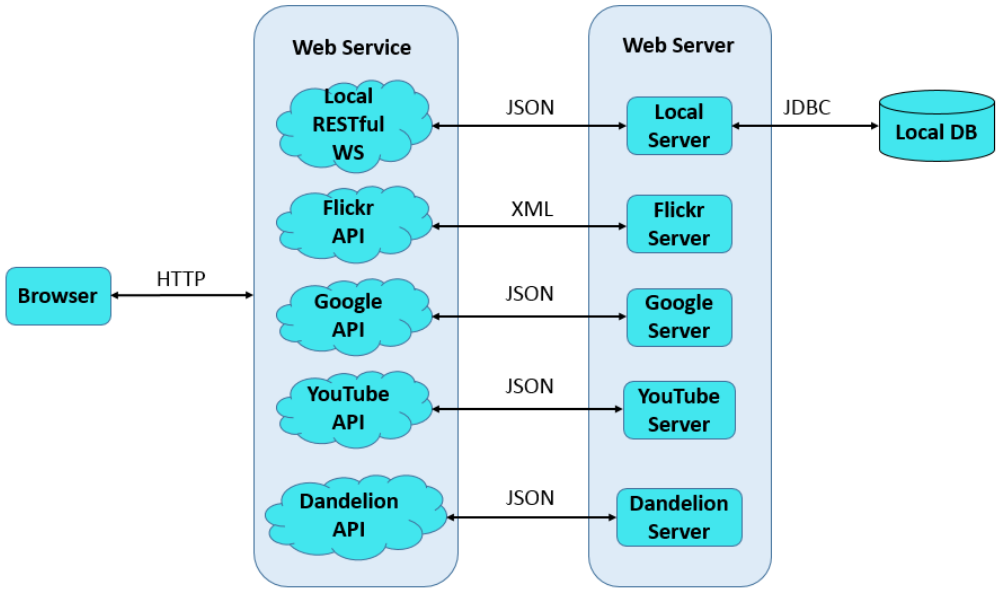


Figure 1: System Architecture Diagram

The web-based client can get information from the web server through HTTP request and response. From the System Architecture Diagram, we can see that there are 5 web services available in the web-based application: Local RESTful Web Service, Flickr, Google, YouTube and Dandelion. Local RESTful Web Service is used to get and add local movie information with local database; Flickr is used to search pictures; Google Search can provide us google search services; YouTube is used to search videos; Dandelion can give us sentiment analysis service.

4. Database Schema

Table 1: Movies table

Field Name	Type	Data Integrity Constraints	Meaning
id	INT(11)	PK, NN, AI	User ID
title	VARCHAR(255)	-	Movie title
director	VARCHAR(255)	-	Director information
writer	VARCHAR(255)	-	Writer information
stars	VARCHAR(255)	-	Stars information

genres	VARCHAR(255)	-	Genres information
plot	VARCHAR(255)	-	Plot description
rating	DECIMAL(4,2)	-	Rating information
date	VARCHAR(255)	-	Release date
poster	VARCHAR(255)	-	Poster url
url	VARCHAR(255)	-	Video url

Here, the primary key “id” is set to be auto-increment and users are not allowed to add “id” information, which can help to protect accuracy of the database.

5. Deployment Instruction

5.1 Web Configuration Information

- Jndi-name: MovieJNDI;
- Pool-name: mysql_id26346990_fit5192a1Pool.
- Domain Configuration Code:

```
<jdbc-connection-pool allow-non-component-callers="false" associate-with-thread="false" connection-creation-retry-attempts="0" connection-creation-retry-interval-in-seconds="10" connection-leak-reclaim="false" connection-leak-timeout-in-seconds="0" connection-validation-method="auto-commit" datasource-classname="com.mysql.jdbc.jdbc2.optional.MysqlDataSource" fail-all-connections="false" idle-timeout-in-seconds="300" is-connection-validation-required="false" is-isolation-level-guaranteed="true" lazy-connection-association="false" lazy-connection-enlistment="false" match-connections="false" max-connection-usage-count="0" max-pool-size="32" max-wait-time-in-millis="60000" name="mysql_id26346990_fit5192a1Pool" non-transactional-connections="false" pool-resize-quantity="2" res-type="javax.sql.DataSource" statement-timeout-in-seconds="-1" steady-pool-size="8" validate-atmost-once-period-in-seconds="0" wrap-jdbc-objects="false">
    <property name="serverName" value="localhost"/>
    <property name="portNumber" value="3306"/>
    <property name="databaseName" value="id26346990"/>
    <property name="User" value="fit5192a1"/>
    <property name="Password" value=""/>
    <property name="URL" value="jdbc:mysql://localhost:3306/id26346990?zeroDateTimeBehavior=convertToNull"/>
    <property name="driverClass" value="com.mysql.jdbc.Driver"/>
</jdbc-connection-pool>
<jdbc-resource enabled="true" jndi-name="MovieJNDI" object-type="user" pool-name="mysql_id26346990_fit5192a1Pool"/>
```

You should add this code segment into glassfish-4.1\glassfish\domains\domain1\config\domain.xml file, between the <resources> and </resources> tags.

5.2 Import Database Dump

Open MySQL Workbench, and import the database through the following steps:

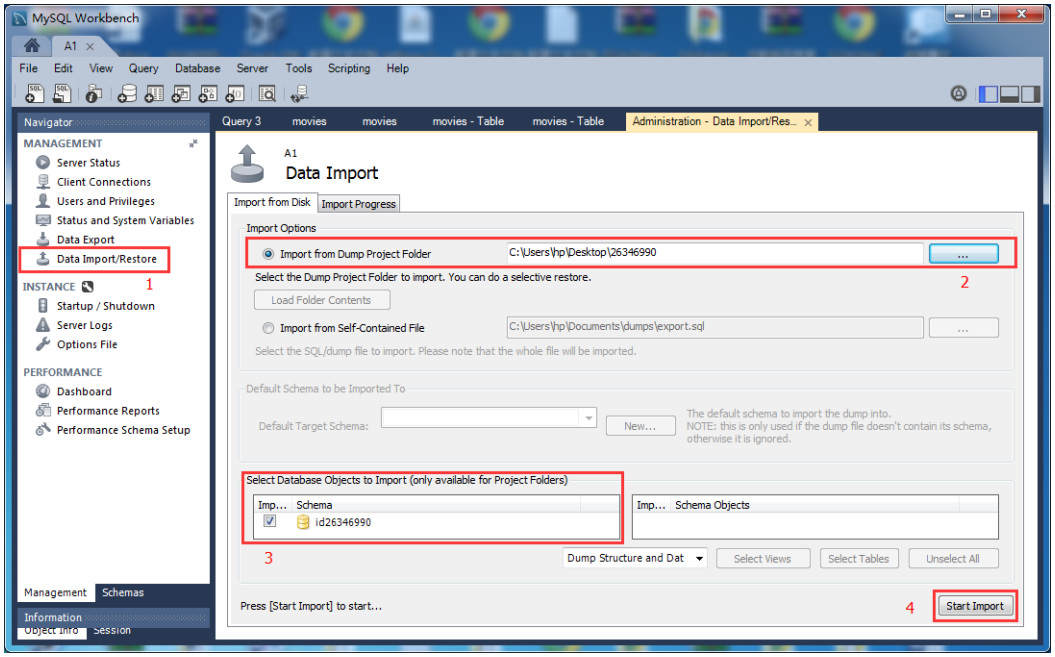


Figure 2: Import database dump

5.3 Deploy the Project

In the \26346990\Dist\ folder, you will find two .war files:

- MovieRestfulWebService.war;
- MovieSearchClient.war

Copy all the two files into the folder:

\glassfish-4.1\glassfish\domains\domain1\autodeploy\

Then, start the glassfish server by starting \glassfish-4.1\glassfish\bin\startserv.bat.

5.4 Application Access Points

- Server-side access point:
  http://localhost:8080/MovieRestfulWebService/webresources/application.wadl
- Client-side access point:
  http://localhost:8080/MovieSearchClient/home.html

6. User Manner

6.1 Home Page

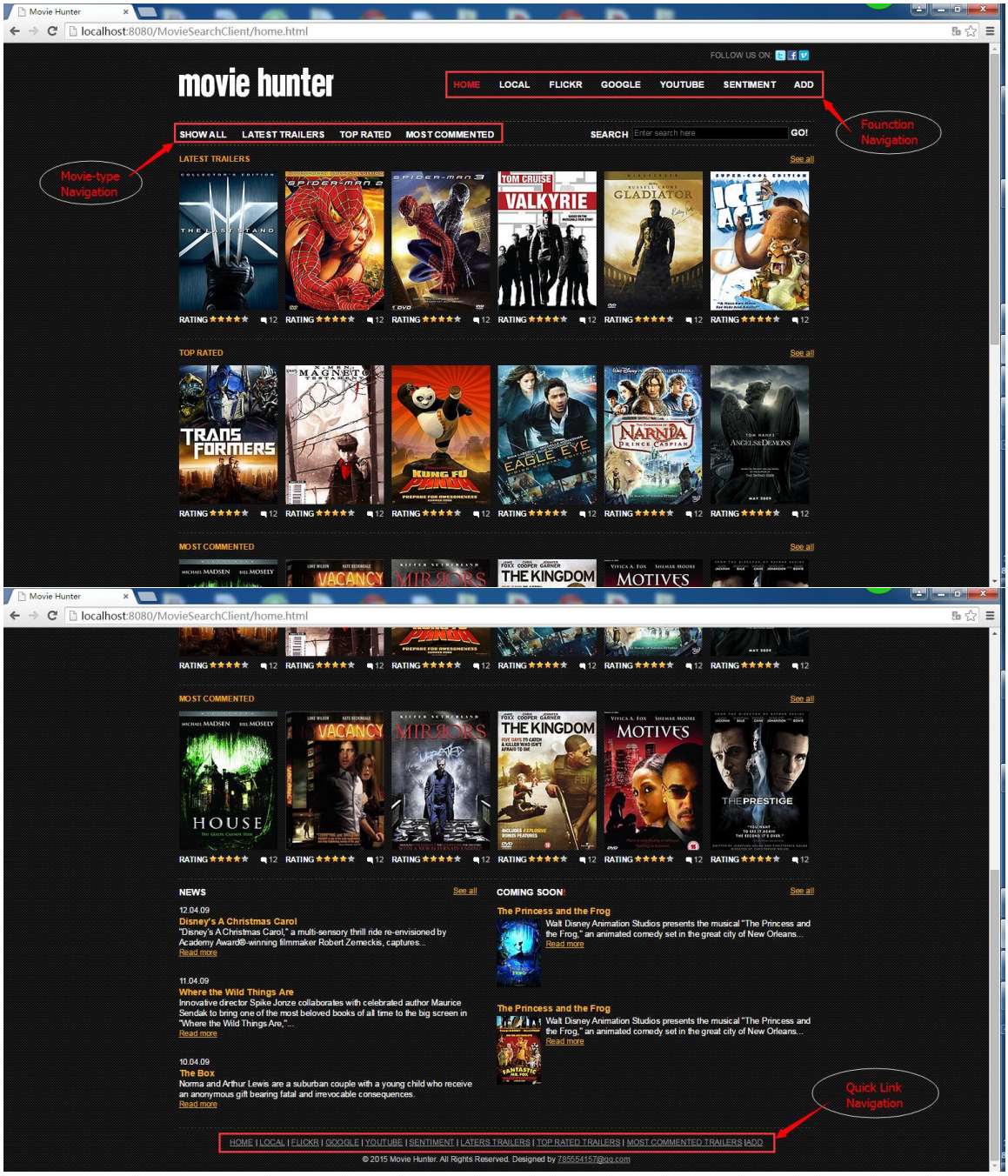


Figure 3: Home Page

As shown in Figure 3, there are function navigation, movie-type navigation and quick-link navigation in the home page. And we can search the movie information from the local database through the search box.

6.2 Local Movie Search Page

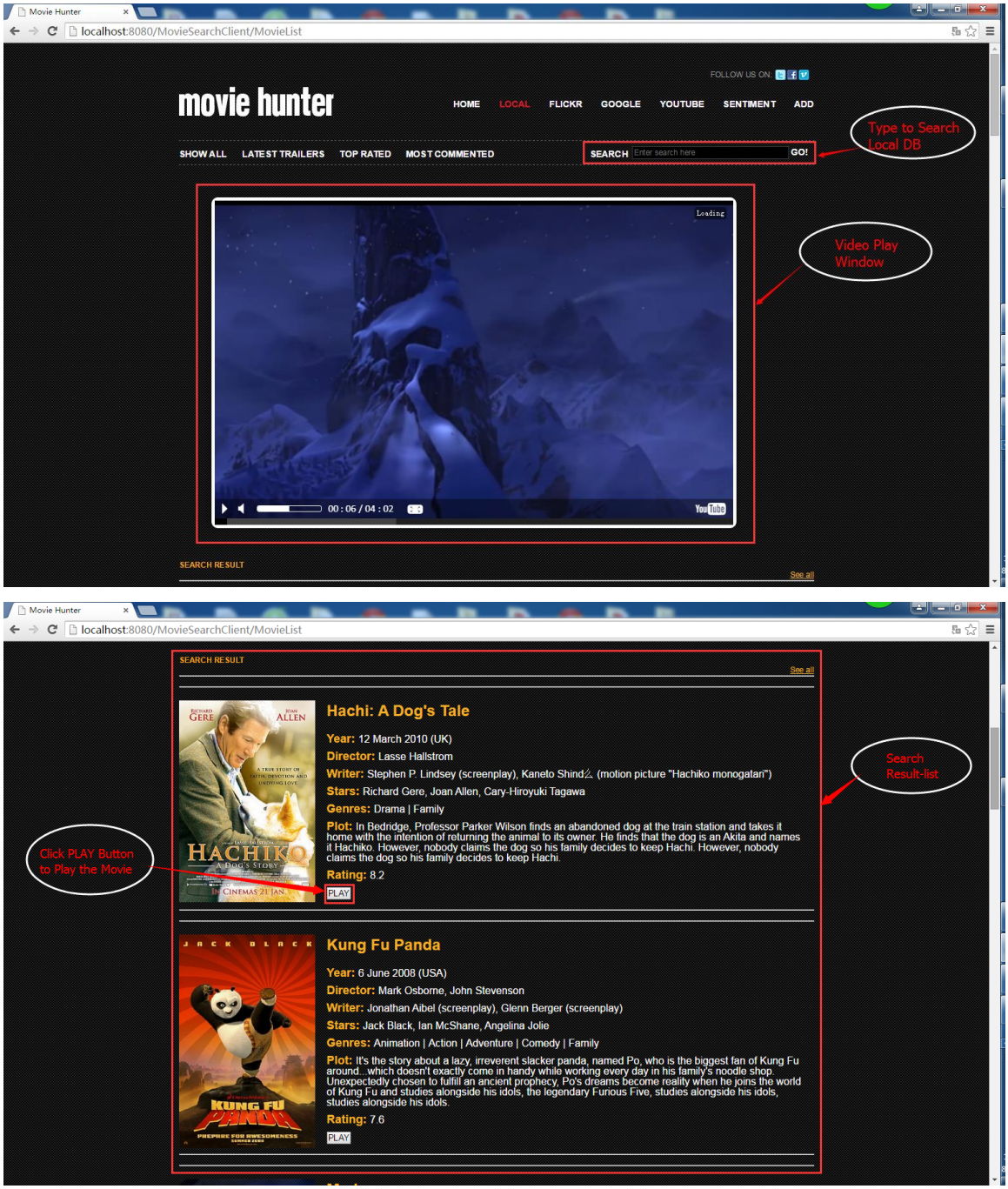


Figure 4: Local Movie Search Page

As shown in Figure 4, there is a video play window in the local movie search page. Below that, there is the search result list and we can click the “PLAY” button to change to play another video. Also, we can search the movie information from the local database through the search box.

6.3 Flickr Picture Search Page

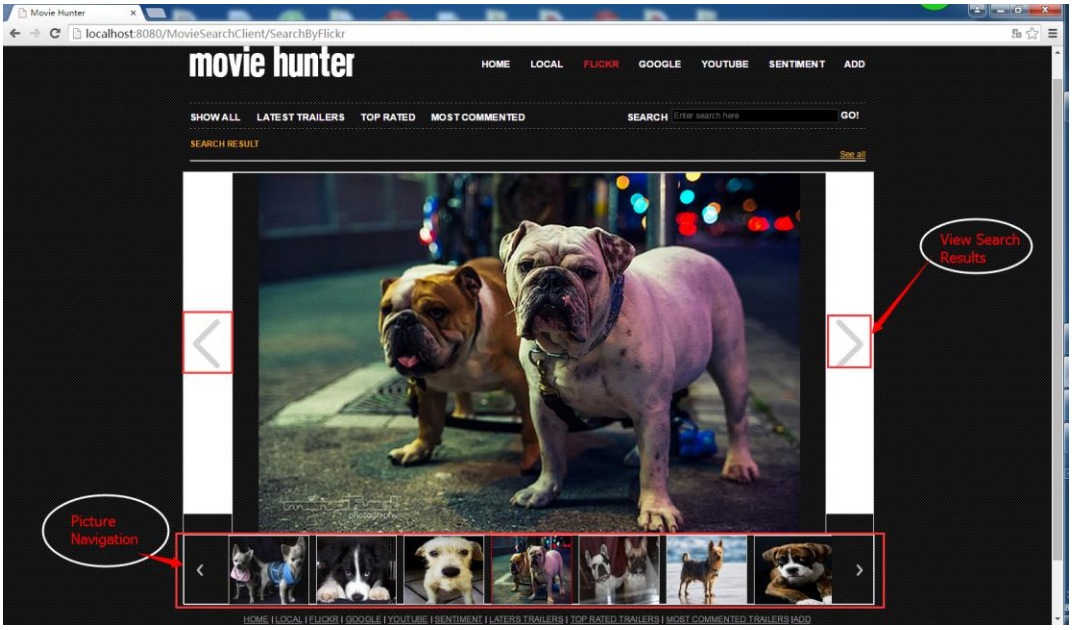


Figure 5: Flickr Picture Search Page

As shown in Figure 5, there is a picture view window in the Flickr picture search page. Below that, there is the Picture Navigation and we can click the navigation buttons to change to show another picture. Also, we can search pictures from the Flickr server through the search box.

6.4 Google Search Page

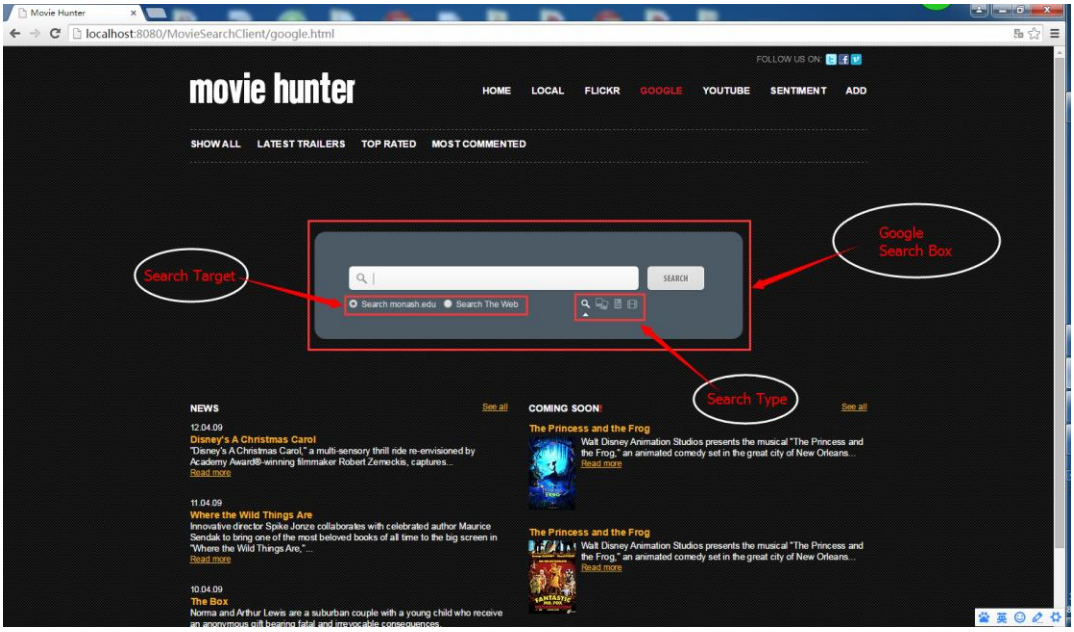


Figure 6: Google Search Page

As shown in Figure 6, there is a google search box in the middle of the google search page. Here, we can choose the search target (search from the target website or from the whole web), and the search type (search websites, pictures, news or videos). Below the search box is the result list, and there are 4 kinds of result list corresponding to the 4 search types:

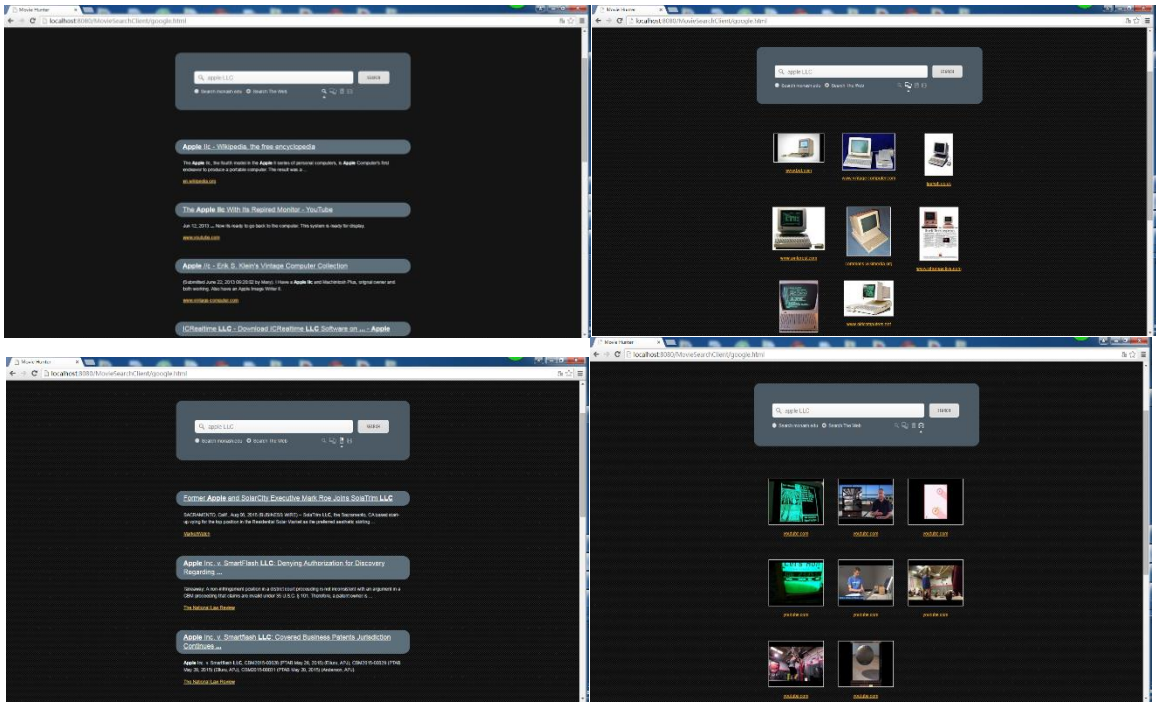
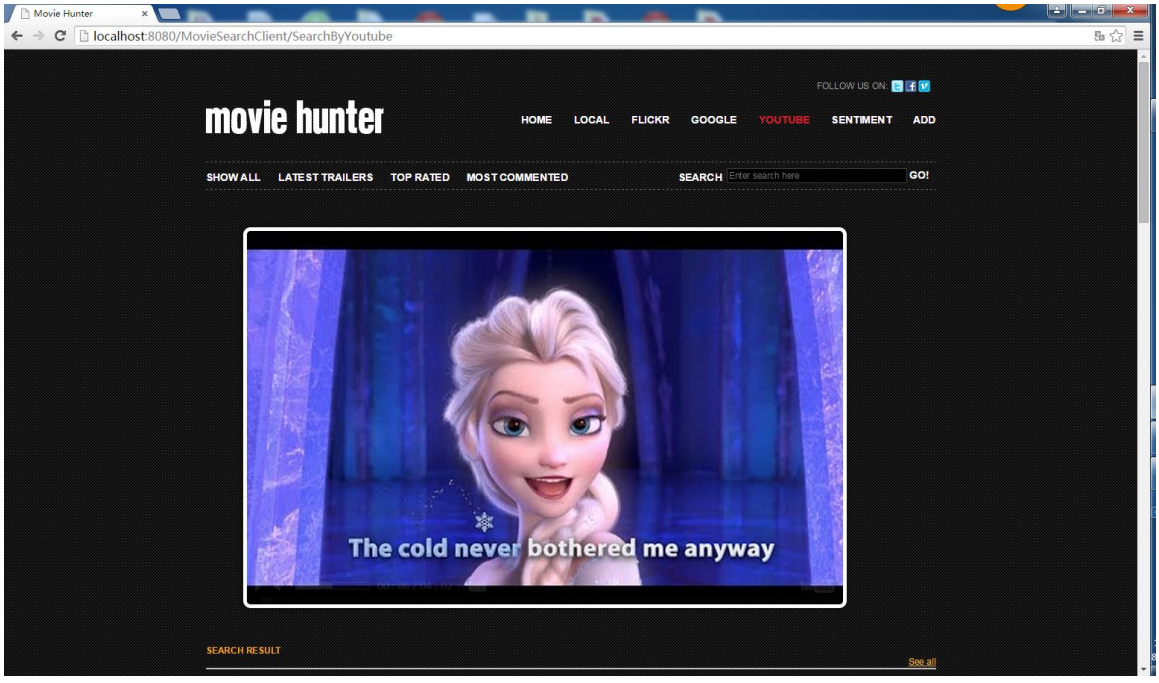


Figure 6: Google Search Results

6.5 YouTube Video Search Page



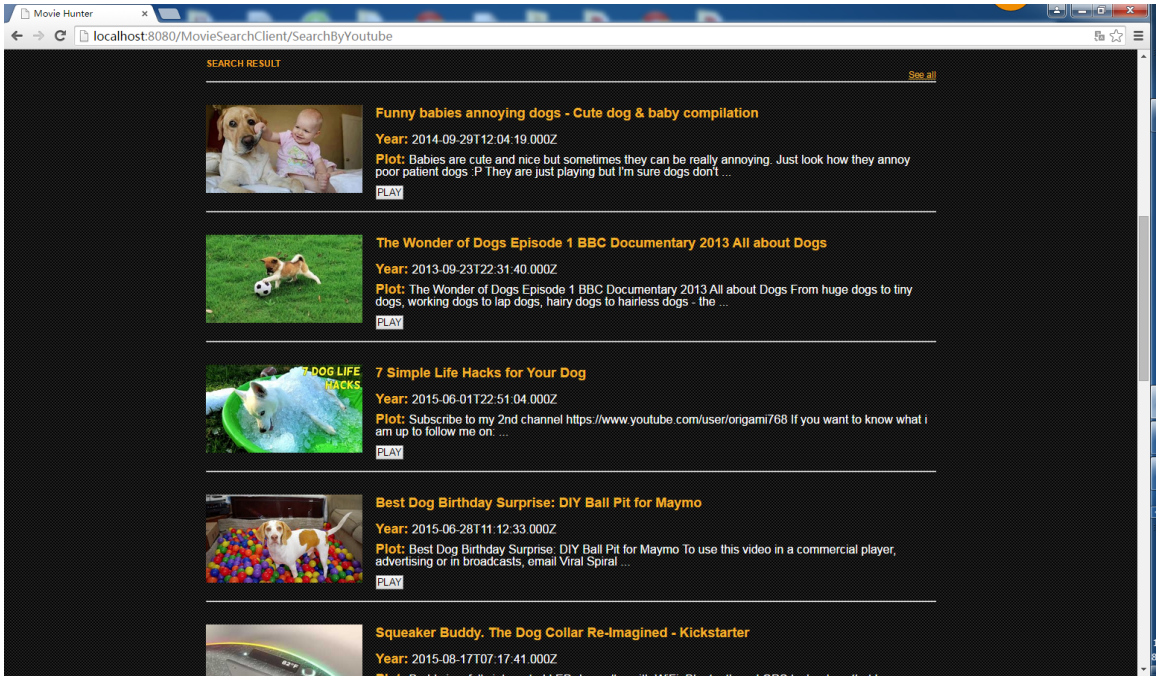


Figure 7: YouTube Video Search Page

Figure 7 shows the YouTube video search page, which is similar to the local movie search page, so we can simply pass this part.

6.6 Dandelion Sentiment Analysis Page

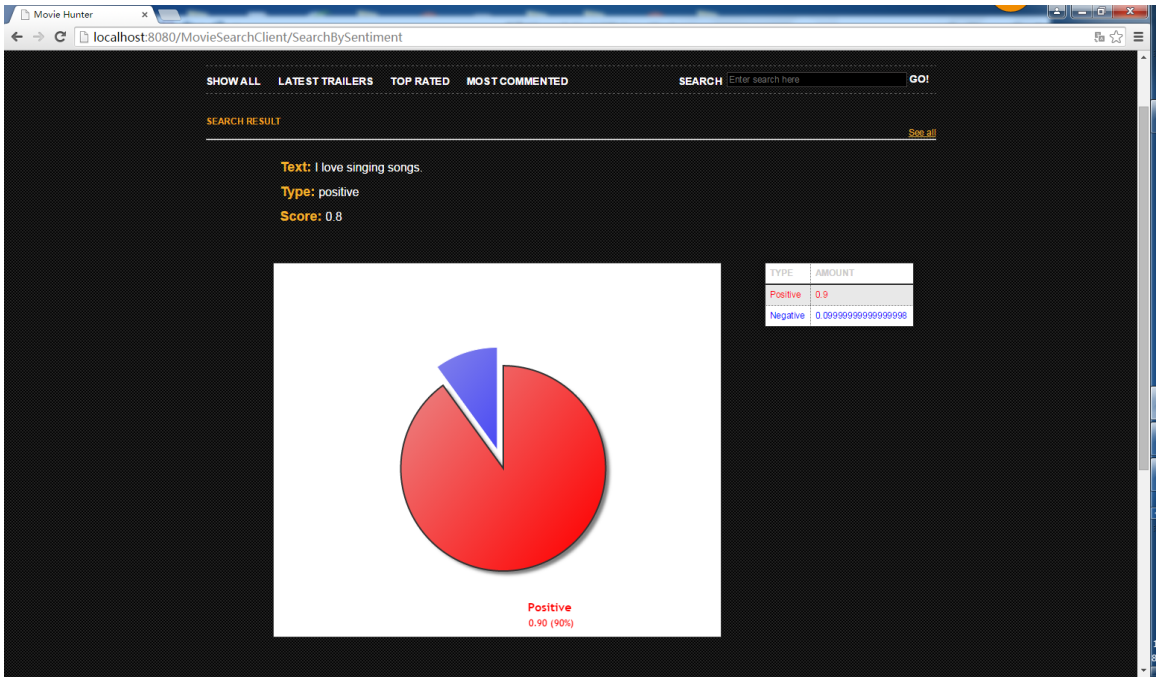


Figure 8: Dandelion Sentiment Analysis Page

Figure 8 shows the Dandelion sentiment analysis page, which can tell whether the expressed opinion in short texts is positive, negative, or neutral. Type in and search a short sentence, it returns a label representing the identified sentiment, along with a numeric score ranging from strongly positive (1.0) to extremely negative (-1.0). The pie chart can present the percentage of the positive and the negative.

6.7 Add Movie Form

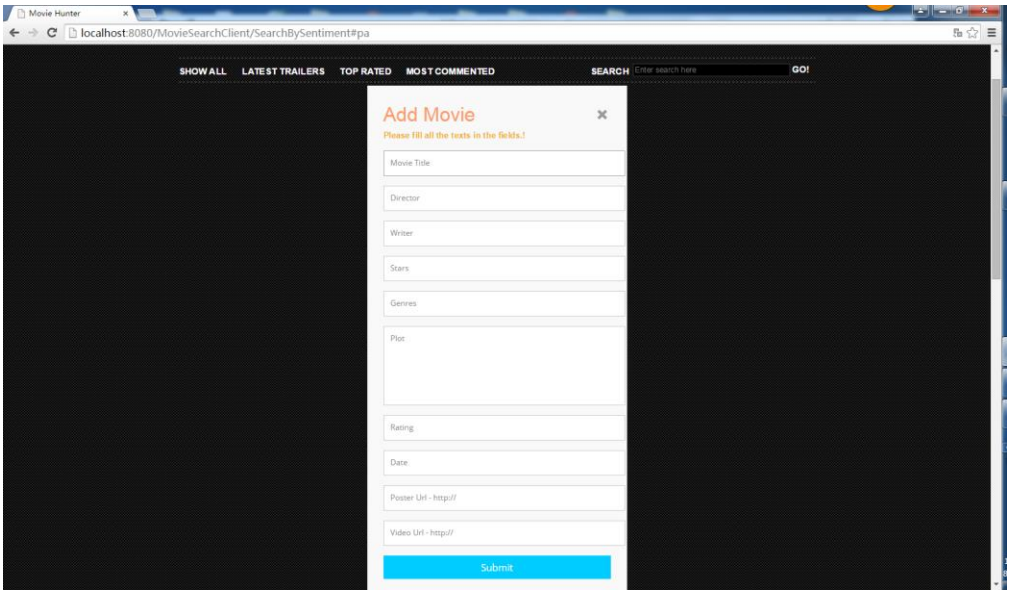


Figure 9: Add Movie Form

Whenever you click the “ADD” label, it will always popup the add movie form, which is as shown in Figure 9. We can fill out this form to add movie information into the local database. The add movie form also provides data validation:

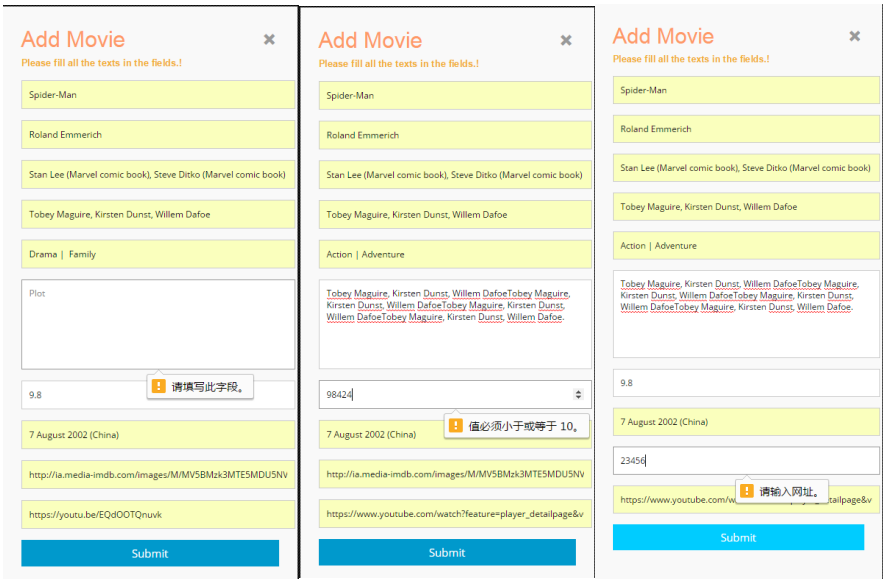


Figure 10: Data Validation

6.8 Error Page

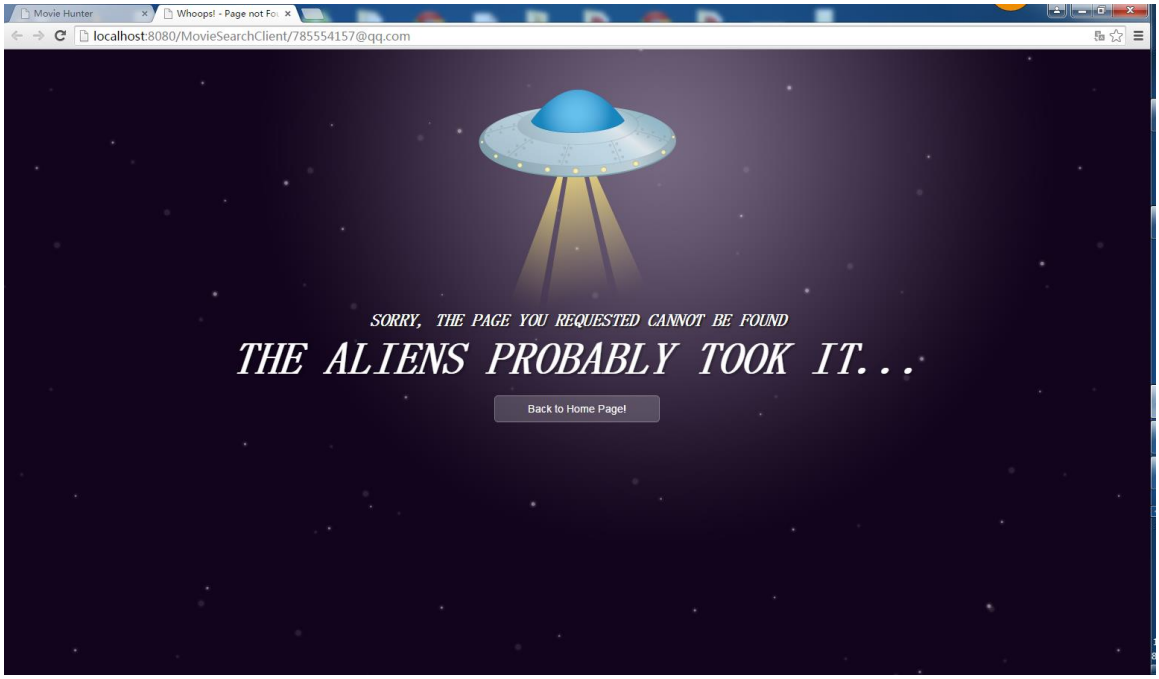


Figure 11: Error Page

When this page appears, whoops --- there must be something wrong! Fortunately, you still can click the button to go back to the home page.

7. Enhancement

- User-friendly UI;
- Exception handling & error page proper usage:

  When exception occurs, the browser does not only show some confusing error information, instead, it will forward to the error page and allow users to go back to the home page.

  We can configure the error page by adding the follow code segment into the web.xml file between <web-app> and </web-app> tags.

```
<error-page>
        <error-code>400</error-code>
        <location>/error.html</location>
</error-page>

<error-page>
        <error-code>404</error-code>
        <location>/error.html</location>
</error-page>

<error-page>
        <error-code>500</error-code>
        <location>/error.html</location>
</error-page>

<error-page>
        <exception-type>javax.servlet.ServletException</exception-type>
        <location>/error.html</location>
</error-page>
```

- Data Validation:

  HTML 5 has provide basic data validation function in <input> tag by adding "type" attribute. We can use this feature to realize data validation function.

```
<fieldset>
    <input name="rating" placeholder="Rating" type="number" min="0" max="10" step="0.1" tabindex="8" required>
</fieldset>
<fieldset>
    <input name="date" placeholder="Date" type="text" tabindex="9" required>
</fieldset>
<fieldset>
    <input name="poster" placeholder="Poster Url - http://" type="url" tabindex="10" required>
</fieldset>
```

  - Type=”number” min="0" max="10" step="0.1" means the input type must be number, it should between 0 and 10. The decimal digits must be 1.
  - The “required” attribute means that the input must not be null.
  - The type="url" means that the input must be a valid url.
  - Besides, there are also “email”, “tell” types, which can also provide data validation function.