

Movie Review Website

Chitra Paryani | Web Development | 27/04/2018

CONTENTS

PROJECT SUMMARY-----1

FUNCTIONALITY PERFORMED-----1

TECHNOLOGIES USED-----2

ROLES AND TASKS-----3

SCREENSHOTS-----4

APPENDIX-----13

PROJECT SUMMARY

- I have created a Spring MVC Hibernate Application in which user can search for movies, see movie reviews and add favorite movies to their collection
- Integrated the application with TMDB API to fetch Movie Data
- There are basically 3 roles – General role, User role and Admin role
- In general, any user can search for movies and see movie reviews
- User needs to login to like the movie and to add it in their favorite collection
- Admin can see all the users and have authorization to delete any user

FUNCTIONALITY PERFORMED

- Integrated with TMDB API to fetch Movie Data
- Displaying All the currently running movies to all the Users
- All Users can see Movie Details/ Movie Reviews
- All Users can search for any Movie and see Movie Reviews
- User needs to login if User wants to like movie and add movie in its collection.
- Created User Registration form for User Logging in for the first time
- Created User Login form to login into the website
- Validations are performed so that duplicate users are not allowed
- User can view his/her own details, update details if needed using its own profile
- Once User logs in, User can see his/her profile, its own Movie Collection
- Movies which are liked by User are added in MyMovies folder
- User can retrieve movie details from their own collection also
- User can Unlike, liked movie incase if user wants to do so
- If User is admin, User can see list of all the Users
- Admin can delete any user incase if it requires to do so
- Admin can download report in either pdf/xls format
- For web security added servlet filters which intercepts, sanitize and clean every request and response on my web application
- Added Basic AUTH mechanism provided by Spring Security for providing additional security

TECHNOLOGIES USED

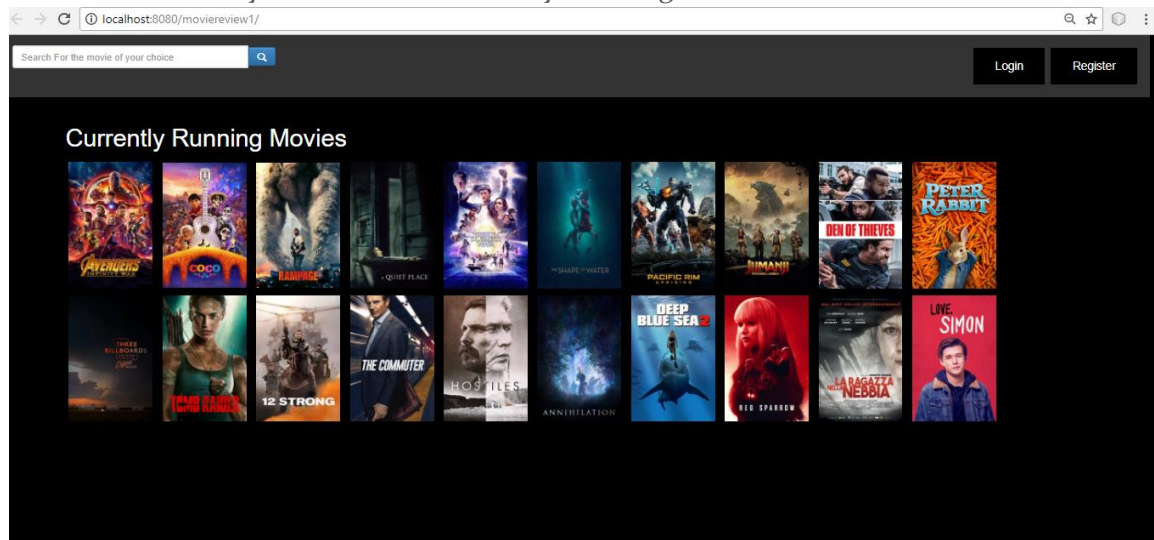
- Spring MVC
- Hibernate
- JSP
- MAVEN
- JSON
- AJAX
- TMDB API
- HTML
- CSS
- JavaScript
- Bootstrap
- MySQL

KEY ROLES AND TASK LIST

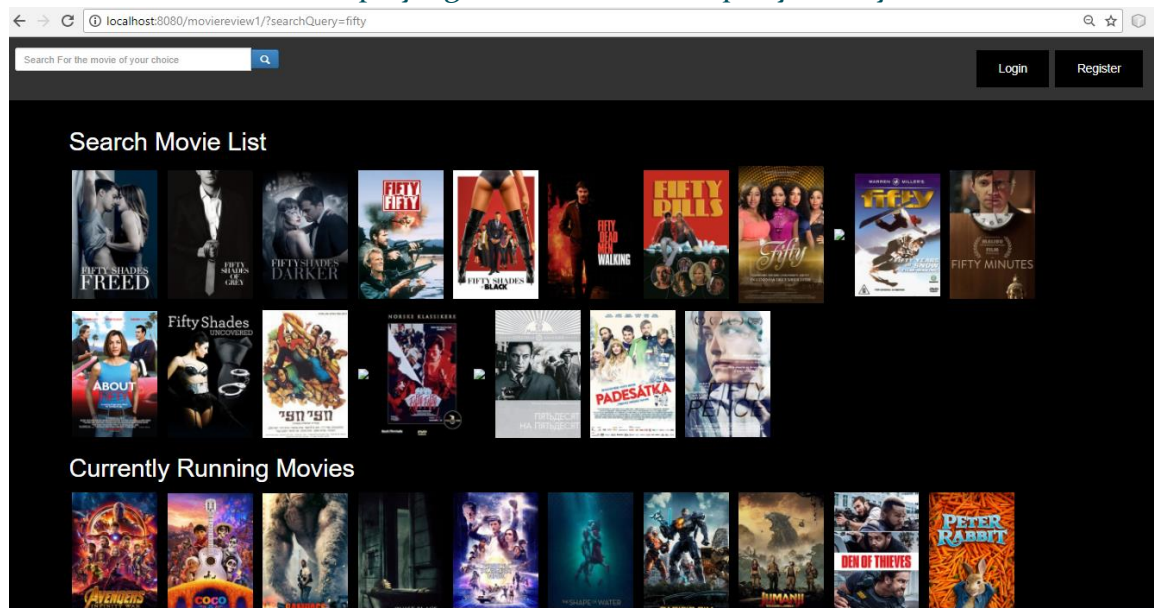
- General Role – Any user can hit URL search and see movie reviews
- User Role – User needs to login to like any movie and to add it in its own collection. User can update his own profile.
- Admin Role – Admin can view all the users and delete any user if it is required to do so. Admin can also view reports and download it either in xls or pdf format

SCREENSHOTS

HOME PAGE – Any User can see Currently Running Movies and Search for Movies



SEARCH RESULTS – Displaying results for search query – “fifty”




MOVIE REVIEWS – User can click on any of the above images to see movie reviews and user needs to log in if user wants to like movies and add it in its own collection

Movie Reviews

Coco

The celebration of a lifetime



Despite His Family'S Baffling Generations-Old Ban On Music, Miguel Dreams Of Becoming An Accomplished Musician Like His Idol, Ernesto De La Cruz. Desperate To Prove His Talent, Miguel Finds Himself In The Stunning And Colorful Land Of The Dead Following A Mysterious Chain Of Events. Along The Way, He Meets Charming Trickster Hector, And Together, They Set Off On An Extraordinary Journey To Unlock The Real Story Behind Miguel's Family History.

Release Date 2017-10-27

Rating: 7.8/10


Popularity 283.283854

Please login to like Movies and to add it in your Collection

REGISTRATION FORM – User needs to register if logging in for the first time

← → ↻ ⓘ localhost:14723/movieview1/user?query=&type=Register 🔍 ⚙️ ⋮

Registration for New User



PIRATES OF THE CARIBBEAN 5

First Name

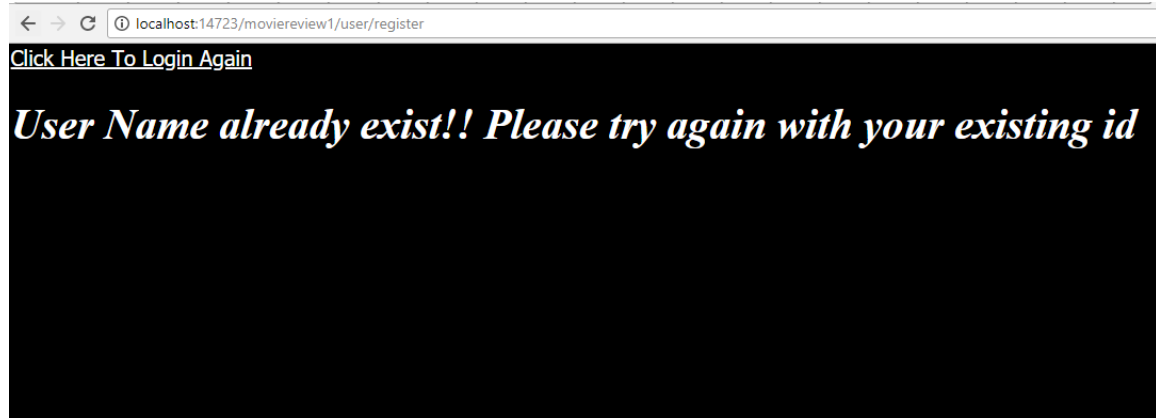
Last Name

Your Email

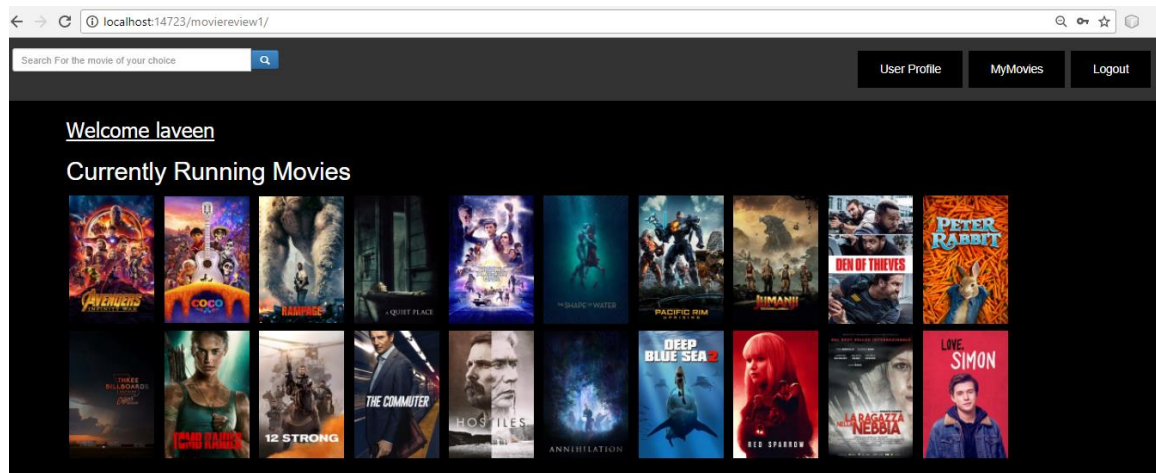
Username

Password

Proper Validations are applied so that no two duplicate users can be created



Once registration is completed, User needs to login and after user logs in, user sees Welcome message, its own profile and its own movie folder



Now, User can click on any of its favorite movie to like movie

Movie Reviews

Avengers: Infinity War

An entire universe. Once and for all.



As The Avengers And Their Allies Have Continued To Protect The World From Threats Too Large For Any One Hero To Handle, A New Danger Has Emerged From The Cosmic Shadows: Thanos. A Despot Of Intergalactic Infamy, His Goal Is To Collect All Six Infinity Stones, Artifacts Of Unimaginable Power, And Use Them To Inflict His Twisted Will On All Of Reality. Everything The Avengers Have Fought For Has Led Up To This Moment - The Fate Of Earth And Existence Itself Has Never Been More Uncertain.

Release Date 2018-04-25

Rating: 8.8/10

Popularity 288.878908



Like button
displayed as heart
icon using

Now, User can see liked movie in its My Movies Collection and directly click from here to view movie reviews again and unlike it also if user wishes to do so

Your Movie Collection - Iaveen



Movie unlike by User – Laveen

Movie Reviews

Avengeers: Infinity War

An entire universe. Once and for all.



As The Avengers And Their Allies Have Continued To Protect The World From Threats Too Large For Any One Hero To Handle, A New Danger Has Emerged From The Cosmic Shadows: Thanos. A Despot Of Intergalactic Infamy, His Goal Is To Collect All Six Infinity Stones, Artifacts Of Unimaginable Power, And Use Them To Inflict His Twisted Will On All Of Reality. Everything The Avengers Have Fought For Has Led Up To This Moment - The Fate Of Earth And Existence Itself Has Never Been More Uncertain.

Release Date 2018-04-25

Rating: 8.8/10

Popularity: 288.878908



Empty heart shows that user has unlike this movie

Now, User collection includes only 2 movies

Your Movie Collection - laveen



User Profile – In this page user can view its own profile and edit details if required

P.SNote- User cannot change its userid and updating it is set to disabled

← → ↻ ⓘ localhost:14723/movieview1/user-profile 🔍 ☆

Edit Profile

First Name
laveen

Last Name
Paryani

Your Email
laveen.paryani@gmail.com

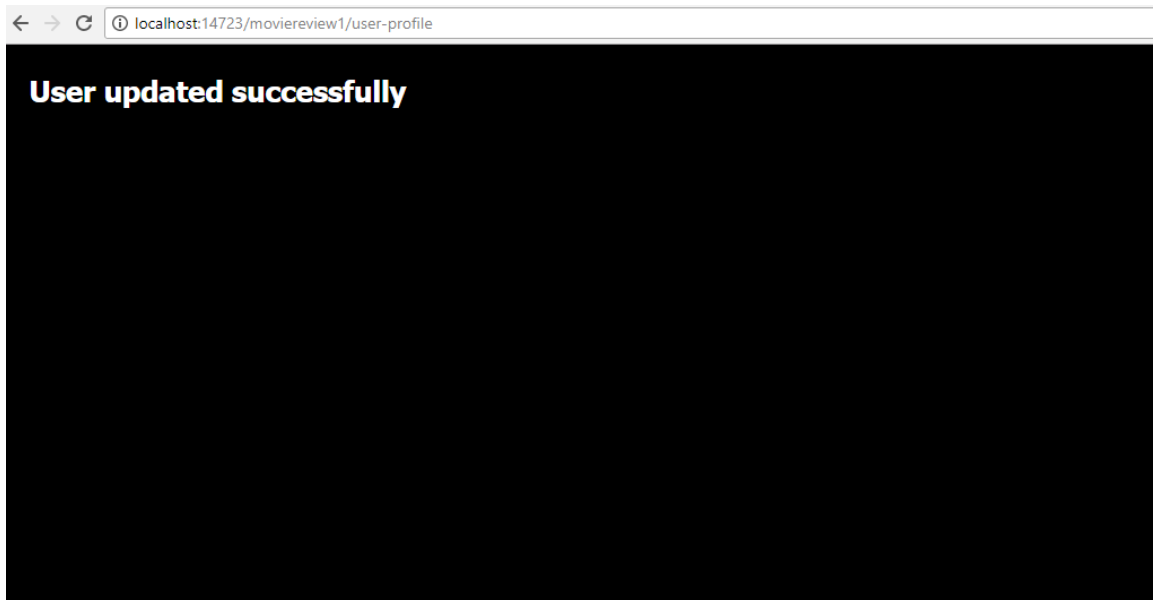
Username
laveenp

Password

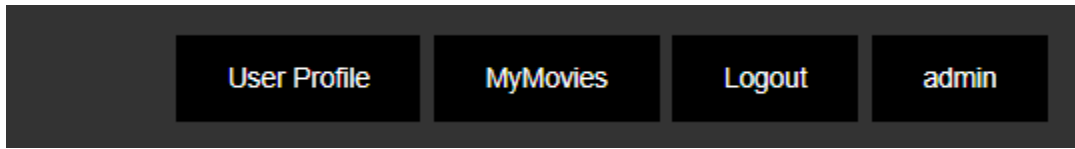
Update

Back

Receives success message once user details updated successfully



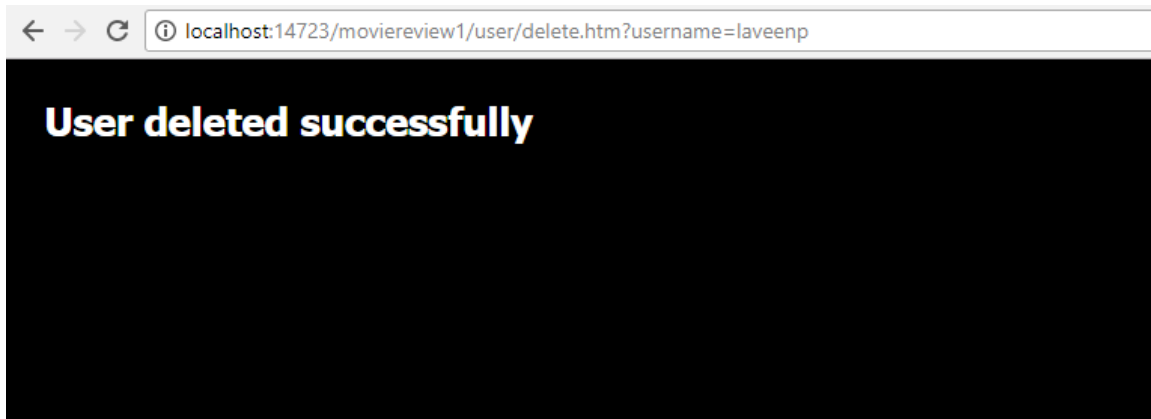
When Admin logs in – admin sees additional Admin page based on its role and task list



Admin can view all the users who logged in the Movie Review Web Site and delete them if needed. User can download user report in either excel or pdf format

				Excel	PDF
UserName	FirstName	LastName			
priyap	pr	paryani			×
sonip	soni	paryani			×
ramchandp	ramchand	paryani			×
gautamp	gautam	vashist			×
ankity	ankit	yadav			×
zibiav	zibia	vadakoot			×
poori	poornima	bansode			×
soniparyani	soni	soni			×

User Deleted By Admin



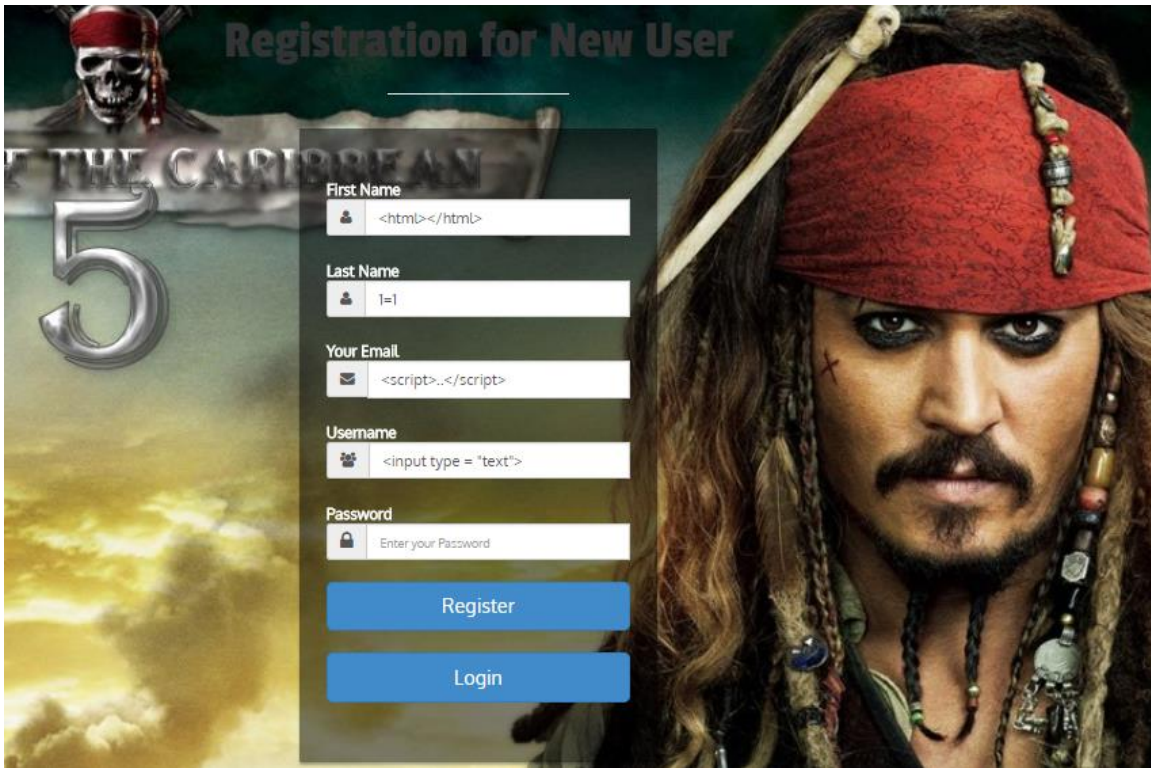
Excel View

A	B	C
username	firstName	lastName
priyap	pr	paryani
sonip	soni	paryani
ramchandp	ramchand	paryani
gautamp	gautam	vashist
ankity	ankit	yadav
zibiav	zibia	vadakoot
poori	poornima	bansode
admin	priya	paryani
soniparyani	soni	soni
<html>	<html>	<html>
html	html	html

PDF View

User Name: priyap
First Name: pr
Last Name: paryani
User Name: sonip
First Name: soni
Last Name: paryani
User Name: ramchandp
First Name: ramchand
Last Name: paryani
User Name: gautamp
First Name: gautam
Last Name: vashist
User Name: ankity
First Name: ankit
Last Name: yadav

Added filters to sanitize, clean and filter data to prevent Cross-Site Scripting
Example below



Registration for New User

5 OF THE CARIBBEAN

First Name
[icon] <html></html>

Last Name
[icon] 1=1

Your Email
[icon] <script>..</script>

Username
[icon] <input type = "text">

Password
[icon] Enter your Password

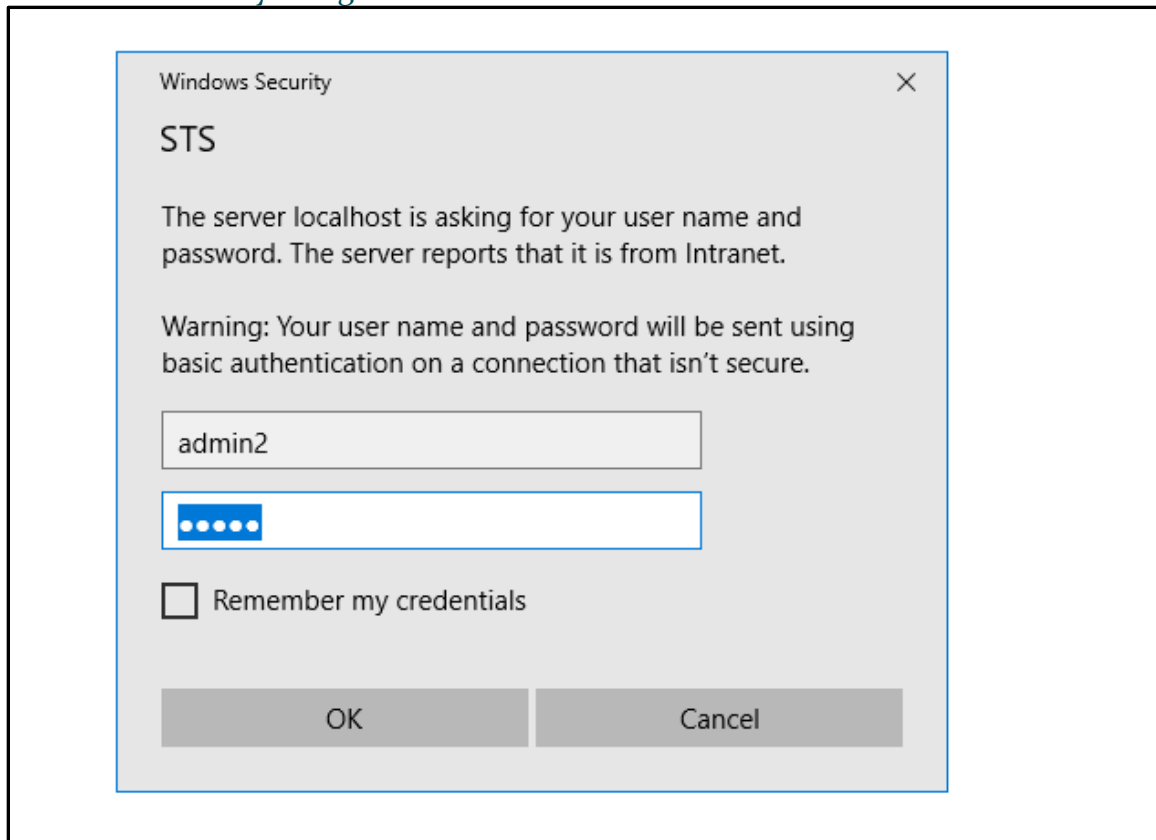
Register

Login

While adding to database, data is filtered and sanitized

25	script .. script	html	html	1 1	cxxmx..	input type...
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Additional security using BASIC form authentication



Ajax

Working on Ajax to speedily retrieve movies from TMDB API

```
<script type="text/javascript" charset="utf-8">
  var api_key = '3e8655395a80eddd5854026d790f304a';

  $(document).ready(function(){
    $.ajax({
      url: 'http://api.themoviedb.org/3/search/movie?api_key=' + api_key + '&query=' + movName,
      dataType: 'json',
      jsonpCallback: 'testing'
    }).error(function() {
      console.log('error')
    }).done(function(response) {
      for (var i = 0; i < response.results.length; i++) {
        $('#search_results').append('<li>' + response.results[i].title + '</li>');
      }
    });
  });
</script>
```


APPENDIX

CONTROLLER SOURCE CODES

MovieController

```
package com.my.moviereview1;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.Reader;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.nio.charset.Charset;
import java.util.HashMap;
import java.util.Map.Entry;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.json.JSONArray;
import org.json.JSONObject;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
```

```

import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.servlet.ModelAndView;


@Controller

public class MovieController {


    private static final Logger logger =
LoggerFactory.getLogger(MovieController.class);


    @RequestMapping(value = "/", method = RequestMethod.GET)

    public ModelAndView displayMovies(HttpServletRequest request,
HttpServletRequest response, ModelMap model) {


        ModelAndView mvc = null;

        for (Entry<String, Object> entry : model.entrySet()) {

            System.out.println("Key = " + entry.getKey() + ", Value = " +
entry.getValue());

        }

        String key = "3e8655395a8oeddd5854026d79of304a";

        String urlBase =
"https://api.themoviedb.org/3/movie/now_playing?api_key="+ key + "&language=en-US";

```

```
String urlSearch = "https://api.themoviedb.org/3/search/movie?api_key="
+ key + "&query=";
```

```
URL url;
```

```
URLConnection con;
```

```
BufferedReader bf;
```

```
String output;
```

```
String mov = request.getParameter("searchQuery");
```

```
try {
```

```
    //collect urlbase in url
```

```
    url = new URL(urlBase);
```

```
    //open connection
```

```
    con = (URLConnection) url.openConnection();
```

```
    con.setDoOutput(true);
```

```
    con.setRequestMethod("GET");
```

```
    con.setRequestProperty("Content-Type", "application/json");
```

```
    bf = new BufferedReader(new
InputStreamReader(con.getInputStream()));
```

```
    while((output = bf.readLine()) !=null) {
```

```
        System.out.println("++++++++++++++++I am
output");
```

```
        System.out.println(output);
```

```

    }

    JSONObject json = readJsonFromUrl(urlBase);

    System.out.println("I am jsonobject" +json.toString());

    JSONArray movieArray = json.getJSONArray("results");

    model.addAttribute("movieArray", movieArray);

    for(int i = 0; i < movieArray.length(); i++) {

        //System.out.println("Json length" +movieArray.length());

        JSONObject obj = movieArray.getJSONObject(i);

        System.out.println("Movie Title      " +obj.get("title"));

    }

    mvc = new ModelAndView("movies", "movieArray", movieArray);

} catch (MalformedURLException e) {

    // TODO Auto-generated catch block

    System.out.println("MalformedURLException" + e.getMessage());

    e.printStackTrace();

} catch(IOException e) {

    System.out.println("IO Exception" +e.getMessage());

    e.printStackTrace();

}

```

```

if(mov != null)
{
try {

//collect urlbase in url

urlSearch = urlSearch + mov;

url = new URL(urlSearch);


//open connection

con = (URLConnection) url.openConnection();

con.setDoOutput(true);

con.setRequestMethod("GET");

con.setRequestProperty("Content-Type", "application/json");


bf = new BufferedReader(new
InputStreamReader(con.getInputStream()));

while((output = bf.readLine()) !=null) {

System.out.println("++++++++++++++++I am

output");

System.out.println(output);

}

JSONObject json = readJsonFromUrl(urlSearch);

System.out.println("I am jsonobject" +json.toString());

```

```

JSONArray movieSArray = json.getJSONArray("results");

model.addAttribute("movieSArray", movieSArray);

for(int i = 0; i < movieSArray.length(); i++) {
    //System.out.println("Json length" +movieArray.length());
    JSONObject obj = movieSArray.getJSONObject(i);
    System.out.println("Movie Title      " +obj.get("title"));
}

//mvc = new ModelAndView("movies", "movieArray", movieArray);

} catch (MalformedURLException e) {
    // TODO Auto-generated catch block
    System.out.println("MalformedURLException" + e.getMessage());
    e.printStackTrace();
} catch(IOException e) {
    System.out.println("IO Exception" +e.getMessage());
    e.printStackTrace();
}

HashMap<String,Object> hm= new HashMap<String,Object>();
//hm.put(", arg1)
}

```



```

        return mvc;
    }

```

```

private static String readAll(Reader rd) throws IOException {
    StringBuilder sb = new StringBuilder();
    int cp;
    while((cp=rd.read())!=-1) {
        sb.append((char)cp);
    }
    System.out.println("I am StringBuilder" +sb.toString());
    return sb.toString();
}

```

```

public static JSONObject readJsonFromUrl(String url) throws
MalformedURLException, IOException {
    InputStream ip = new URL(url).openStream();
    BufferedReader br1 = new BufferedReader(new InputStreamReader(ip,
Charset.forName("UTF-8")));
    String jsonString = readAll(br1);
    JSONObject json = new JSONObject(jsonString);
    return json;
}

```

```

//@RequestMapping(value = "/", method = RequestMethod.GET)

```

```
        public String displayProfile(HttpServletRequest request) {  
  
            return null;  
        }  
    }  
}
```

MovieDetailsController

```
package com.my.moviereview1;  
  
import java.io.BufferedReader;  
import java.io.IOException;  
import java.io.InputStream;  
import java.io.InputStreamReader;  
import java.io.Reader;  
import java.net.HttpURLConnection;  
import java.net.MalformedURLException;  
import java.net.URL;  
import java.nio.charset.Charset;  
  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import javax.servlet.http.HttpSession;  
  
import org.json.JSONArray;
```

```

import org.json.JSONObject;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.servlet.ModelAndView;


import com.my.moviereview1.dao.UserDAO;

import com.my.moviereview1.pojo.UserMovies;


@Controller

public class MovieDetailsController {


    private static final Logger logger =
LoggerFactory.getLogger(MovieDetailsController.class);


    @RequestMapping(value = "/movie", method = RequestMethod.GET)

    public ModelAndView displayDetails(HttpServletRequest request,
HttpServletRequest response, UserDAO userDao, ModelMap model) throws
IOException {

        ModelAndView mvc = null;


        String key = "3e8655395a8oeddd5854026d79of304a";

        String movieId = request.getParameter("id");

```

```
String urlBase = "https://api.themoviedb.org/3/movie/" + movieId +  
"?api_key=" + key
```

```
+ "&append_to_response=credits&language=en-US";
```

```
URL url;
```

```
URLConnection con;
```

```
BufferedReader br;
```

```
String output;
```

```
url = new URL(urlBase);
```

```
con = (URLConnection) url.openConnection();
```

```
con.setDoOutput(true);
```

```
con.setRequestMethod("GET");
```

```
con.setRequestProperty("Content-Type", "application/json");
```

```
br = new BufferedReader(new  
InputStreamReader((con.getInputStream())));
```

```
while ((output = br.readLine()) != null) {
```

```
    System.out.println("+++++++I am output");
```

```
    System.out.println(output);
```

```
}
```

```
HttpSession session = request.getSession();
```

```
if(session !=null) {
```

```
    session.getAttribute("user");
```

```
    session.getAttribute("username");
```

```
        model.addAttribute("user");  
        model.addAttribute("username");  
        model.addAttribute(session);  
    }  
  
}
```

```
JSONObject movie = readJsonFromUrl(urlBase);  
System.out.println("I am jsonobject" + movie.toString());  
mvc = new ModelAndView("movie-details", "movie", movie);
```

```
//Storing boolean value to session
```

```
String username = null;  
username = (String) session.getAttribute("username");  
if(username != null) {  
    boolean flag = userDao.retrieveMovieId(username, movieId);  
    //boolean flag = true;  
    session.setAttribute("flag", flag);  
}
```

```
return mvc;
```

```
}
```

```

private static String readAll(Reader rd) throws IOException {

    StringBuilder sb = new StringBuilder();

    int cp;

    while ((cp = rd.read()) != -1) {

        sb.append((char) cp);

    }

    System.out.println("I am StringBuilder" + sb.toString());

    return sb.toString();

}

```

```

public static JSONObject readJsonFromUrl(String url) throws
MalformedURLException, IOException {

    InputStream ip = new URL(url).openStream();

    BufferedReader br1 = new BufferedReader(new InputStreamReader(ip,
Charset.forName("UTF-8")));

    String jsonString = readAll(br1);

    JSONObject json = new JSONObject(jsonString);

    return json;

}

```

```

@RequestMapping(value = "/movie-details", method = RequestMethod.GET)

public String likeClick(HttpServletRequest request, UserDao userDao,
UserMovies um) throws Exception {

```



```

String key = "3e8655395a80eddd5854026d79of304a";

String movieId = request.getParameter("id");

System.out.println(movieId);

String userName = null;

HttpSession session = request.getSession();

if(session !=null) {

    session.getAttribute("user");

    userName = (String) session.getAttribute("username");

}

if(!userDao.retrieveMovieId(userName, movieId))

    um = userDao.saveUser(userName, movieId);

else

    userDao.deleteUser(userName, movieId);

return "redirect:/movie/?id=" + movieId;

}

```

```

//      public String countLike(HttpServletRequest request, UserDao userDao,
UserMovies um, ModelMap model) {

//          HttpSession session = request.getSession();

//          String username = null;

//          if(session !=null) {

//              username = (String) session.getAttribute("username");

//          }

```

```
//          String id = request.getParameter("id");  
//          boolean flag = userDao.retrieveMovieId(username, id);  
//          model.addAttribute("flag", flag);  
//          return "movie-details";  
//      }  
  
}
```

UserAdminController

```
package com.my.moviereview1;  
  
import java.util.ArrayList;  
import java.util.List;  
  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import javax.servlet.http.HttpSession;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.beans.factory.annotation.Qualifier;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.ModelMap;  
import org.springframework.web.bind.annotation.RequestMapping;
```

```
import org.springframework.web.bind.annotation.RequestMethod;
```

```
import org.springframework.web.servlet.ModelAndView;
```

```
import com.my.moviereview1.dao.UserDAO;
```

```
import com.my.moviereview1.exception.UserException;
```

```
import com.my.moviereview1.pojo.User;
```

```
import com.my.moviereview1.report.ExcelView;
```

```
import com.my.moviereview1.report.PdfView;
```

```
@Controller
```

```
public class UserAdminController {
```

```
    @RequestMapping(value = "/movie-admin", method = RequestMethod.GET)
```

```
    public ModelAndView displayUser(HttpServletRequest request,  
    HttpServletResponse response, UserDAO userDao, ModelMap model) throws  
    UserException {
```

```
        HttpSession session = request.getSession(false);
```

```
        ArrayList<User> userlist = new ArrayList<User>();
```

```
        if(session.getAttribute("userid").equals("admin")) {
```

```
            userlist =  
            userDao.displayUser((String)session.getAttribute("userid"));
```

```
        }
```

```
        return new ModelAndView("movie-admin", "userlist", userlist);
```

```
}
```

```
@RequestMapping(value = "/user/delete.htm", method = RequestMethod.GET)  
  
public ModelAndView deleteUser(HttpServletRequest request, UserDao userDao)  
throws Exception {
```

```
    ModelAndView mv = null;  
  
    String id = request.getParameter("username");  
  
    userDao.deleteUserByAdmin(id);  
  
    mv = new ModelAndView("user-deleted");  
  
    return mv;  
  
}
```

```
@RequestMapping(value = "/movie-admin", method = RequestMethod.POST)  
  
protected ModelAndView saveCSV(HttpServletRequest request,  
HttpServletResponse response, UserDao userDao) {
```

```
    ModelAndView mv = null;  
  
    ArrayList<User> list = (ArrayList<User>) userDao.retrieveUser();  
  
    mv = new ModelAndView(new ExcelView(), "list", list);  
  
    return mv;  
  
}
```

```

    @RequestMapping(value = "/movie-admin/pdf",method = RequestMethod.GET)

    protected ModelAndView savePDF(HttpServletRequest request,
    HttpServletResponse response, UserDao userDao) {

        ModelAndView mv = null;

        ArrayList<User> list = (ArrayList<User>) userDao.retrieveUser();

        mv = new ModelAndView(new PdfView(), "list", list);

        return mv;

    }

}

```

UserController

```

package com.my.moviereview1;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import javax.xml.ws.Response;

```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.InitBinder;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.servlet.ModelAndView;
```

```
import com.my.moviereview1.exception.UserException;
import com.my.moviereview1.dao.UserDAO;
import com.my.moviereview1.pojo.User;
import com.my.moviereview1.pojo.UserMovies;
import com.my.moviereview1.validator.UserValidator;
```

```
@Controller
```

```
public class UserController {
```

```
    @Autowired
```

```
    @Qualifier("userDao")
```

```
    UserDAO userDao;
```



```

@Autowired

@Qualifier("userValidator")
UserValidator validator;

@InitBinder

private void initBinder(WebDataBinder binder) {

    binder.setValidator(validator);

}

@RequestMapping(value = "/user", method = RequestMethod.GET)

public ModelAndView loginForm(HttpServletRequest request,
    HttpServletResponse response, ModelMap model, @ModelAttribute("user") User user)
    throws IOException {

    String value = request.getParameter("type");

    //String action = request.getParameter("action");

    ModelAndView mvc = null;

    if (value.equals("Login")) {

        mvc = new ModelAndView("user-form");

    } else if (value.equals("Register")) {

        mvc = new ModelAndView("register-user", "user", new User());

    } else if (value.equals("Logout")) {

        HttpSession session = request.getSession();

        if (session != null) {

            session.invalidate();

        }

    }

}

```

```

        response.sendRedirect("/moviereview1/");
    }

} else if (value.equals("User Profile")) {
    HttpSession session = request.getSession(false);
    String username = (String) session.getAttribute("username");
    if (session != null) {
        if (session.getAttribute("username") != null) {
            //      List<UserMovies> list =
userDao.retrieveMovieUser(username);

            //      model.addAttribute("list", list);

            //mvc = new ModelAndView("user-profile", "list",
list);

            response.sendRedirect("/moviereview1/user-
profile");

        }
    }
}

} else if (value.equals("admin")) {
    HttpSession session = request.getSession(false);
    if (session != null) {
        String userid = (String) session.getAttribute("userid");
        if (userid.equals("admin")) {
            session.setAttribute("userid", userid);
            response.sendRedirect("/moviereview1/movie-
admin");

        }
    }
}

```

```

    }
}
else if (value.equals("MyMovies")) {
    HttpSession session = request.getSession(false);
    String username = (String) session.getAttribute("username");
    if (session != null) {
        if (session.getAttribute("username") != null) {
            response.sendRedirect("/moviereview1/user-
movies");

            //mvc = new ModelAndView("user-movies");
        }
    }
}
else if(value.equals("search")) {
    String searchQuery = request.getParameter("query");
    System.out.println(searchQuery);
    model.addAttribute("searchQuery", searchQuery);

    mvc = new
ModelAndView("redirect:/", "searchQuery", searchQuery);

}

return mvc;
}

```

```

@RequestMapping(value = "/user/login", method = RequestMethod.GET)
protected ModelAndView userForm() throws Exception {
    System.out.print("User Logged In");
    return new ModelAndView("user-form");
}

```

```
}
```

```
@RequestMapping(value = "/user/login", method = RequestMethod.POST)

protected String userLogin(HttpServletRequest request, ModelMap model) throws
Exception {

    System.out.print("User Logged In");

    HttpSession session = (HttpSession) request.getSession();

    String userid = request.getParameter("username");

    String password = request.getParameter("password");

    //String cancel = request.getParameter("Cancel");

    User u = null;

    try {

        System.out.print("loginUser");

        u = userDao.get(userid,password);

        if(u == null ) {

            System.out.println("UserName/Password does not
exist");

            model.addAttribute("errorMessage",
"UserName/Password does not exist");

            return "error";

        } else if (!(u.getUsername().equals("userid") ||
(u.getPassword().equals(password))))

        {

            System.out.println("UserName/Password
does not exist");
```

```

        model.addAttribute("errorMessage",
"UserName/Password does not exist");

        return "error";

    }

    session.setAttribute("user", u);

    session.setAttribute("username", u.getFirstName());

    session.setAttribute("userid", u.getUsername());

    //return "redirect:./movies";

} catch (UserException e) {

    System.out.println("Exception: " + e.getMessage());

    session.setAttribute("errorMessage", "error while login");

    return "error";

}

//ModelAndView mvc = new ModelAndView("movies",
"movieArray", model.get("movieArray"));

return "redirect:/";

}

@RequestMapping(value = "/user/register", method = RequestMethod.GET)
protected ModelAndView registerUser() throws Exception {

    System.out.print("Register New User");

    return new ModelAndView("register-user", "user", new User());

}

```

```

    @RequestMapping(value = "/user/register", method = RequestMethod.POST)

    protected String registerNewUser(HttpServletRequest request,
    @ModelAttribute("user") User user,

        BindingResult result) throws Exception {

        System.out.print("Register New User");

        validator.validate(user, result);

        if (result.hasErrors()) {

            return "error";

        }

        ArrayList<User> userlist= (ArrayList<User>)
userDao.retrieveUser();

        String usernam= user.getUsername();

        for(User u: userlist) {

            if(usernam.equals(u.getUsername())) {

                request.getSession().setAttribute("errorMessage",
"User Name already exist!! Please try again with your existing id");

                return "error";

            }

        }

        HttpSession session = request.getSession(false);

        System.out.print("registerNewUser");

```

```
        User u = userDao.register(user);

        request.getSession().setAttribute("user", u);

        session.setAttribute("user", u);

        session.setAttribute("username", u.getFirstName());

        session.setAttribute("userid", u.getUsername());

        return "redirect:/";

    }

}
```

UserMoviesController

```
package com.my.moviereview1;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.Reader;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.nio.charset.Charset;
import java.util.ArrayList;
import java.util.List;
```

```
import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;


import org.json.JSONArray;
import org.json.JSONObject;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.InitBinder;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.ModelAndView;


import com.my.moviereview1.dao.UserDAO;
import com.my.moviereview1.pojo.User;
import com.my.moviereview1.pojo.UserMovies;
import com.my.moviereview1.validator.UserValidator;


@Controller

public class UserMoviesController {
```



```

@RequestMapping(value = "/user-movies", method = RequestMethod.GET)

public ModelAndView displayDetails(HttpServletRequest request,
HttpServletResponse response, UserDao userDao,

        ModelMap model) throws IOException {

    ModelAndView mvc = null;

    String key = "3e8655395a8oeddd5854026d79of304a";

    JSONArray ja = new JSONArray();

    HttpSession session = request.getSession(false);

    String username = null;

    ArrayList movies = new ArrayList();

    List<UserMovies> list = new ArrayList<UserMovies>();

    if (session != null) {

        username = (String) session.getAttribute("username");

        session.getAttribute("user");

        if (username != null) {

            list = userDao.retrieveMovieUser(username);

        }

        if (list.size() <= 0) {

            mvc = new ModelAndView("user-movies", "message",
"Currently No Movies Added!!!");

        }
    }

```

```

        for (UserMovies um : list) {

            String movieid = um.getMovieid();

            movies.add(movieid);

            String urlBase = "https://api.themoviedb.org/3/movie/" +
movieid + "?api_key=" + key
                                +
"&append_to_response=credits&language=en-US";

            URL url;
            HttpURLConnection con;
            BufferedReader br;
            String output;
            url = new URL(urlBase);

            con = (HttpURLConnection) url.openConnection();
            con.setDoOutput(true);
            con.setRequestMethod("GET");
            con.setRequestProperty("Content-Type",
"application/json");

            br = new BufferedReader(new
InputStreamReader((con.getInputStream())));

            while ((output = br.readLine()) != null) {

                System.out.println("+++++++I am
output");

                System.out.println(output);

```

```

    }

    JSONObject movie = readJsonFromUrl(urlBase);

    System.out.println("I am jsonobject" + movie.toString());

    ja.put(movie);

    // System.out.println("I am inside list" +movieid);

}

for (int i = 0; i < ja.length(); i++) {

    // System.out.println("Json length" +movieArray.length());

    JSONObject obj = ja.getJSONObject(i);

    System.out.println("Movie Title      " + obj.get("title"));

}

if (list.size() > 0) {

    mvc = new ModelAndView("user-movies", "ja", ja);

}

}

return mvc;

}

private static String readAll(Reader rd) throws IOException {

    StringBuilder sb = new StringBuilder();

```

```

        int cp;

        while ((cp = rd.read()) != -1) {

            sb.append((char) cp);

        }

        System.out.println("I am StringBuilder" + sb.toString());

        return sb.toString();

    }

```

```

    public static JSONObject readJsonFromUrl(String url) throws
    MalformedURLException, IOException {

        InputStream ip = new URL(url).openStream();

        BufferedReader br1 = new BufferedReader(new InputStreamReader(ip,
        Charset.forName("UTF-8")));

        String JsonString = readAll(br1);

        JSONObject json = new JSONObject(JsonString);

        return json;

    }

```

```

    @RequestMapping(value = "/user-profile", method = RequestMethod.GET)

    public ModelAndView displayUserDetails(HttpServletRequest request,
    HttpServletResponse response, @ModelAttribute("user") User user, UserDao userDao)
    throws Exception {

```

```

        HttpSession session = request.getSession(false);

        if(session!=null) {

```

```

        String username = (String)session.getAttribute("username");

        String userid = (String) session.getAttribute("userid");

        user = userDao.getUser(userid);

    }

```

```

    return new ModelAndView("user-profile", "user", user);

```

```

}

```

```

@RequestMapping(value = "/user-profile", method = RequestMethod.POST)

public ModelAndView EditDetails(HttpServletRequest request,
@ModelAttribute("user") User user,

```

```

        UserDao userDao) throws Exception {

```

```

    HttpSession session = request.getSession(false);

```

```

    String username = null;

```

```

    if(session!=null) {

```

```

        username = (String)session.getAttribute("userid");

```

```

    }

```

```

    ModelAndView mv = null;

```

```

    String name = (String)session.getAttribute("userid");

```

```

    User u = userDao.getId(name);

```

```

    int id = u.getUserId();

```

```

    User user1 = new User();

```

```

    user1.setFirstName(request.getParameter("firstname"));

```

```

        String n = request.getParameter("firstname");
        String surname = request.getParameter("lastName");
        user1.setLastName(request.getParameter("lastname"));
        user1.setPassword(request.getParameter("password"));
        user1.setEmail(request.getParameter("email"));
        user1.setUserId(id);
        user1.setUsername(username);
        int count = userDao.putId(user1);
        mv = new ModelAndView("user-updated");
        return mv;
    }
}

```

```

package com.my.moviereview1.dao;

```

```

import java.util.logging.Level;
import java.util.logging.Logger;

import org.hibernate.HibernateException;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

```

```

public class DAO {

    private static final Logger log = Logger.getAnonymousLogger();

    private static final ThreadLocal sessionThread = new ThreadLocal();

    private static final SessionFactory sessionFactory = new
Configuration().configure("hibernate.cfg.xml").buildSessionFactory();

    protected DAO() {
    }

    public static Session getSession()
    {
        Session session = (Session) DAO.sessionThread.get();

        if (session == null)
        {
            session = sessionFactory.openSession();

            DAO.sessionThread.set(session);
        }

        return session;
    }
}

```

```

protected void begin() {
    getSession().beginTransaction();
}

protected void commit() {
    getSession().getTransaction().commit();
}

protected void rollback() {
    try {
        getSession().getTransaction().rollback();
    } catch (HibernateException e) {
        log.log(Level.WARNING, "Cannot rollback", e);
    }

    try {
        getSession().close();
    } catch (HibernateException e) {
        log.log(Level.WARNING, "Cannot close", e);
    }

    DAO.sessionThread.set(null);
}

public static void close() {
    getSession().close();

    DAO.sessionThread.set(null);
}

```



```
}
```

UserDAO

```
package com.my.moviereview1.dao;
```

```
import java.util.ArrayList;
```

```
import java.util.List;
```

```
import org.hibernate.HibernateException;
```

```
import org.hibernate.Query;
```

```
import com.my.moviereview1.exception.UserException;
```

```
import com.my.moviereview1.pojo.User;
```

```
import com.my.moviereview1.pojo.UserMovies;
```

```
public class UserDAO extends DAO{
```

```
    public UserDAO() {
```

```
    }
```

```
    //Register User
```

```

public User register(User u)
    throws UserException {
    try {
        begin();
        System.out.println("inside DAO");

        User user = new User();
        user.setUsername(u.getUsername());
        user.setPassword(u.getPassword());
        user.setFirstName(u.getFirstName());
        user.setLastName(u.getLastName());
        user.setEmail(u.getEmail());
        getSession().save(user);
        commit();
        return user;

    } catch (HibernateException e) {
        rollback();
        throw new UserException("Exception while creating user: " +
e.getMessage());
    }
}

//get username, password
public User get(String username, String password) throws UserException {
    try {

```

```

        begin();

        Query q = getSession().createQuery("from User where username =
:username and password = :password");

        q.setString("username", username);
        q.setString("password", password);
        User user = (User) q.uniqueResult();
        commit();

        return user;
    } catch (HibernateException e) {

        rollback();

        throw new UserException("Could not get user " + username, e);
    }
}

```

//save user

```

public UserMovies saveUser(String username, String movieid) throws Exception {
    try {
        begin();

        UserMovies um = new UserMovies();
        um.setUsername(username);
        um.setMovieid(movieid);
        getSession().save(um);
        commit();

        return um;
    }catch(HibernateException e) {

        rollback();
    }
}

```

```

        throw new Exception("Could not get user" + username, e);
    }
}

//delete user

public void deleteUser(String username, String movie) throws Exception {
    try {
        Query query = getSession().createQuery("Delete from UserMovies
where username = :username and movieid = :movie");

        query.setParameter("username", username);
        query.setParameter("movie", movie);

        query.executeUpdate();

    }catch(HibernateException e) {
        rollback();
        throw new Exception("Could not get user" + username, e);
    }
}

//Retrive MovieID

public boolean retrieveMovieId(String username, String movie) {

    Query query = getSession().createQuery("from UserMovies where
username = :username and movieid = :movie");

    query.setParameter("username", username);

```

```
query.setParameter("movie", movie);
```

```
List list = query.list();
```

```
if(list.size() > 0) {  
    return true;  
}else {  
    return false;  
}
```

```
}
```

```
public List retrieveMovieUser(String username) {  
    Query query = getSession().createQuery("from UserMovies where  
username = :username");  
    query.setParameter("username", username);  
    List list = query.list();  
    close();  
    return list;  
}
```

```
//List of Users
```

```
public List retrieveUser() {  
    Query query = getSession().createQuery("from User");  
    List list = query.list();
```

```

        close();

        return list;
    }

//Admin

public ArrayList<User> displayUser(String username) throws UserException {

    String name = "admin";

    ArrayList<User> userlist = new ArrayList<User>();

    try {

        begin();

        Query q = null;

        if(username.equals("admin")) {

            q = getSession().createQuery("from User where username !=
:name");

            q.setParameter("name", name);

        }

        userlist = (ArrayList<User>) q.list();

        commit();

        return userlist;

    } catch (HibernateException e) {

        rollback();

        throw new UserException("Could not get user " + username, e);

    }

}

```

```

//delete user

public void deleteUserByAdmin(String username) throws Exception {

    try {

        Query query = getSession().createQuery("Delete from User where
username = :username");

        query.setParameter("username", username);

        query.executeUpdate();

    }catch(HibernateException e) {

        rollback();

        throw new Exception("Could not get user" + username, e);

    }

}

//get user based on name

public User getUser(String username) throws Exception {

    try {

        begin();

        Query q = getSession().createQuery("from User where username =
:username");

        q.setString("username", username);

        User user = (User) q.uniqueResult();

        commit();

        return user;

    } catch (HibernateException e) {

```

```

        rollback();

        throw new UserException("Could not get user " + username, e);
    }
}

public void EditUserDetails(String username) throws Exception {

    try {

        Query query = getSession().createQuery("Update User where
username = :username");

        query.setParameter("username", username);

        query.executeUpdate();

        commit();

    } catch (HibernateException e) {

        rollback();

        throw new Exception("Could not get user" + username, e);

    }

}

public User getId(String username) throws Exception {

    try {

        begin();

```



```
        Query query = getSession().createQuery("from User where  
username = :username");
```

```
        query.setString("username", username);
```

```
        User u = (User) query.uniqueResult();
```

```
        commit();
```

```
        return u;
```

```
    }catch(HibernateException e) {
```

```
        rollback();
```

```
        throw new Exception("Could not get user" + username, e);
```

```
    }
```

```
}
```

```
public int putId(User user) throws Exception {
```

```
    int userid = user.getUserId();
```

```
    String firstname = user.getFirstName();
```

```
    int result = 0;
```

```
    try {
```

```
        begin();
```

```
        Query query = getSession().createQuery("update User set firstname  
= :firstname " +
```

```
            "where userid = :userid");
```

```
        query.setParameter("userid", userid);
```

```
        query.setParameter("firstname", firstname);
```

```

        result = query.executeUpdate();

        commit();

    }catch(HibernateException e) {

        rollback();

        throw new Exception("Could not get user" + user, e);

    }

    return result;

}

}

```

UserException

```

package com.my.moviereview1.exception;

public class UserException extends Exception {

    public UserException(String message) {
        super(message);
    }

    public UserException(String message, Throwable clause) {
        super(message, clause);
    }

}

```

CrossScriptingFilter

```

package com.my.moviereview1.filter;

import java.io.IOException;

import javax.servlet.Filter;

```

```

import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;

import org.apache.log4j.Logger;

public class CrossScriptingFilter implements Filter {

    private static Logger logger =
Logger.getLogger(CrossScriptingFilter.class);

    private FilterConfig filterConfig;

    @Override

    public void init(FilterConfig filterConfig) throws ServletException {

        // TODO Auto-generated method stub

        this.filterConfig = filterConfig;

    }

    @Override

    public void doFilter(ServletRequest request, ServletResponse response,
FilterChain chain)

        throws IOException, ServletException {

        // TODO Auto-generated method stub

        logger.info("Inlter CrossScriptingFilter .....");

```

```

        chain.doFilter(new RequestWrapper((HttpServletRequest) request),
response);

        logger.info("Outlter CrossScriptingFilter .....");

    }

    @Override
    public void destroy() {
        // TODO Auto-generated method stub
        this.filterConfig = null;
    }

}

```

RequestWrapper

```

package com.my.moviereview1.filter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletRequestWrapper;

import org.apache.log4j.Logger;

public class RequestWrapper extends HttpServletRequestWrapper {

    private static Logger logger = Logger.getLogger(RequestWrapper.class);

```

```

static String allowedChars = "+-0123456789#*";

public RequestWrapper(ServletRequest servletRequest) {
    super((HttpServletRequest)servletRequest);
}

public String[] getParameterValues(String parameter) {
    logger.info("InparameterValues .. parameter .....");
    String[] values = super.getParameterValues(parameter);
    if (values == null) {
        return null;
    }
    int count = values.length;
    String[] encodedValues = new String[count];
    for (int i = 0; i < count; i++) {
        encodedValues[i] = cleanXSS(values[i]);
    }
    return encodedValues;
}

public String getParameter(String parameter) {
    logger.info("Inparameter .. parameter .....");
    String value = super.getParameter(parameter);
    if (value == null) {
        return null;
    }

```

```

    }

    logger.info("Inparameter RequestWrapper ..... value .....");
    return cleanXSS(value);
}

public String getHeader(String name) {
    logger.info("Ineader .. parameter .....");
    String value = super.getHeader(name);
    if (value == null)
        return null;
    logger.info("Ineader RequestWrapper ..... value ....");
    return cleanXSS(value);
}

private String cleanXSS(String value) {
    // You'll need to remove the spaces from the html entities below
    logger.info("InnXSS RequestWrapper ..... + value);
    value = value.replaceAll("<", " ").replaceAll(">", " ");
    value = value.replaceAll("=", " ");
    value = value.replaceAll("/", " ");
    value = value.replaceAll("\\\\(", "& #40;").replaceAll("\\\\)", "&
#41;");
    value = value.replaceAll("'", "& #39;");
    value = value.replaceAll("eval\\\\((.*)\\\\)", "");
    value =
value.replaceAll("[\\\\"\\\\'"][\\"\\\\s]*javascript:(.*)[\\\\"\\\\'"]", "\\\"\\\"");

    value = value.replaceAll("(?i)<script.*?>.??<script.*?>", "");
    value = value.replaceAll("(?i)<script.*?>.??</script.*?>", "");

```

```

        value = value.replaceAll("(?i)<.*?javascript:.*?>.*?</.*?>", "");
        value = value.replaceAll("(?i)<.*?\\s+on.*?>.*?</.*?>", "");
        value = value.replaceAll("<script>", "");
        value = value.replaceAll("</script>", "");
        logger.info("OutnXSS RequestWrapper ..... value ..... " +
value);

        return value;
    }

}

```

ExcelView

```

package com.my.moviereview1.report;

import java.util.List;
import java.util.Map;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.ss.usermodel.Sheet;
import org.apache.poi.ss.usermodel.Workbook;
import org.springframework.web.servlet.view.document.AbstractXlsView;

```

```

import com.my.moviereview1.pojo.User;


public class ExcelView extends AbstractXlsView {

    @Override

    protected void buildExcelDocument(Map<String, Object> model, Workbook
workbook, HttpServletRequest request,

        HttpServletResponse response) throws Exception {

        // TODO Auto-generated method stub


        response.setHeader("Content-Disposition", "attachment; filename=
\"UserList.xls\"");


        List<User> list = (List<User>) model.get("list");
        Sheet sheet = workbook.createSheet("UserList");


        Row header = sheet.createRow(0);
        //header.createCell(0).setCellValue("ID");
        header.createCell(0).setCellValue("username");
        header.createCell(1).setCellValue("firstName");
        header.createCell(2).setCellValue("lastName");


        int rowNum = 1;


        for(User user : list) {

            Row row = sheet.createRow(rowNum++);

```



```

        //row.createCell(0).setCellValue(sales.getId());
        row.createCell(0).setCellValue(user.getUsername());
        row.createCell(1).setCellValue(user.getFirstName());
        row.createCell(2).setCellValue(user.getLastName());
    }

}

}

```

PdfView

```

package com.my.moviereview1.report;

import java.util.List;
import java.util.Map;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.web.servlet.view.document.AbstractPdfView;

import com.lowagie.text.Document;
import com.lowagie.text.Paragraph;
import com.lowagie.text.pdf.PdfWriter;

import com.my.moviereview1.pojo.User;

```

```

public class PdfView extends AbstractPdfView {

    @Override

    protected void buildPdfDocument(Map<String, Object> model, Document
pdfdoc, PdfWriter writer,

        HttpServletRequest request, HttpServletResponse response)
throws Exception {

        // TODO Auto-generated method stub

        List<User> list = (List<User>) model.get("list");

        for(User user : list) {

            pdfdoc.add(new Paragraph("User Name: " +
user.getUsername()));

            pdfdoc.add(new Paragraph("First Name: " +
user.getFirstName()));

            pdfdoc.add(new Paragraph("Last Name: " +
user.getLastName()));

        }

    }

}

```

UserValidator

```

package com.my.moviereview1.validator;

import org.springframework.validation.Errors;
import org.springframework.validation.ValidationUtils;
import org.springframework.validation.Validator;

import com.my.moviereview1.pojo.User;

```

```

public class UserValidator implements Validator {

    @Override

    public boolean supports(Class<?> clazz) {

        // TODO Auto-generated method stub

        return clazz.equals(User.class);

    }

    @Override

    public void validate(Object target, Errors errors) {

        // TODO Auto-generated method stub

        User user = (User) target;

        ValidationUtils.rejectIfEmptyOrWhitespace(errors, "firstName",
"error.invalid.user", "First Name Required");

        ValidationUtils.rejectIfEmptyOrWhitespace(errors, "lastName",
"error.invalid.user", "Last Name Required");

        ValidationUtils.rejectIfEmptyOrWhitespace(errors, "username",
"error.invalid.user", "User Name Required");

        ValidationUtils.rejectIfEmptyOrWhitespace(errors, "password",
"error.invalid.password", "Password Required");

        ValidationUtils.rejectIfEmptyOrWhitespace(errors, "email",
"error.invalid.email",

            "Email Required");

    }

}

```

Hibernate.cfg.xml

```
<?xml version="1.0" encoding="UTF-8"?>
```

```

<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration
DTD//EN"

    "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property
name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
        <property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/moviereview</prope
rty>
        <property name="hibernate.connection.username">root</property>
        <property
name="hibernate.connection.password">Diwali@105</property>
        <property
name="hibernate.dialect">org.hibernate.dialect.MySQL5InnoDBDialect</property>
        <property name="hibernate.hbm2ddl.auto">update</property>
        <property name="hibernate.show_sql">true</property>

        <mapping class="com.my.moviereview1.pojo.User" />
        <mapping class = "com.my.moviereview1.pojo.UserMovies" />

    </session-factory>
</hibernate-configuration>

```

POJOS

User

```

package com.my.moviereview1.pojo;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity

```

```

@Table(name = "usertable")

public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "userid", nullable = false, unique = true)
    private int userId;

    @Column(name = "username")
    private String username;

    @Column(name = "password")
    private String password;

    @Column(name = "firstname")
    private String firstName;

    @Column(name = "lastname")
    private String lastName;

    @Column(name = "email")
    private String email;


    public int getUserId() {
        return userId;
    }

    public void setUserId(int userId) {
        this.userId = userId;
    }
}

```

```
}

public String getUsername() {
    return username;
}

public void setUsername(String username) {
    this.username = username;
}

public String getPassword() {
    return password;
}

public void setPassword(String password) {
    this.password = password;
}

public String getFirstName() {
    return firstName;
}

public void setFirstName(String firstName) {
    this.firstName = firstName;
}

public String getLastName() {
    return lastName;
}

public void setLastName(String lastName) {
    this.lastName = lastName;
}

public String getEmail() {
    return email;
}
```

```
        public void setEmail(String email) {  
            this.email = email;  
        }  
  
    }  
  
}
```

UserMovies

```
package com.my.moviereview1.pojo;  
  
import javax.persistence.Column;  
import javax.persistence.Entity;  
import javax.persistence.GeneratedValue;  
import javax.persistence.GenerationType;  
import javax.persistence.Id;  
import javax.persistence.Table;  
  
@Entity  
@Table(name = "usermovietable")  
public class UserMovies {  
  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    @Column(name = "id", nullable = false, unique = true)  
    private int id;  
    @Column(name = "username")  
    private String username;  
}
```

```

@Column(name = "movieid")
private String movieid;

public int getId() {
    return id;
}

public void setId(int id) {
    this.id = id;
}

public String getUsername() {
    return username;
}

public void setUsername(String username) {
    this.username = username;
}

public String getMovieid() {
    return movieid;
}

public void setMovieid(String movieid) {
    this.movieid = movieid;
}

}

```