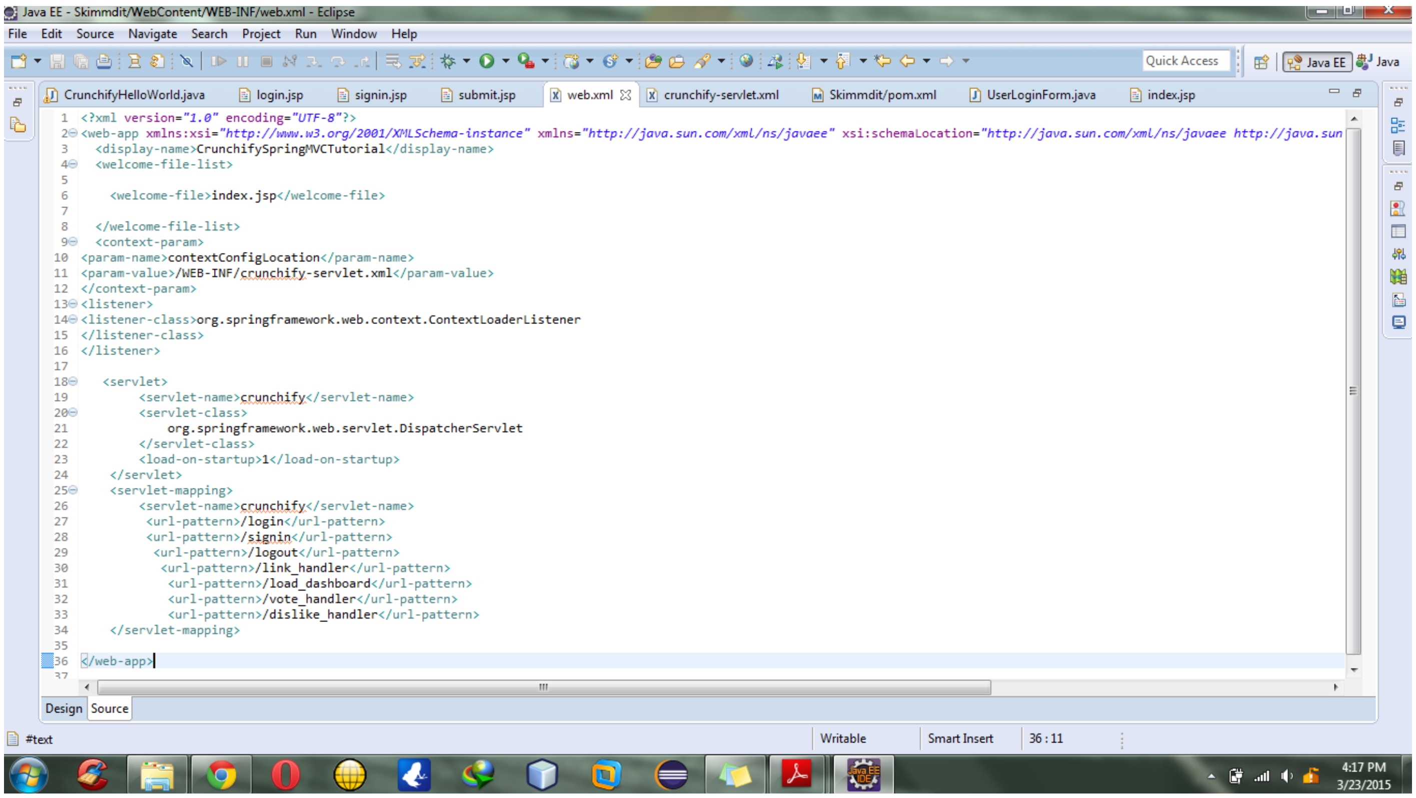
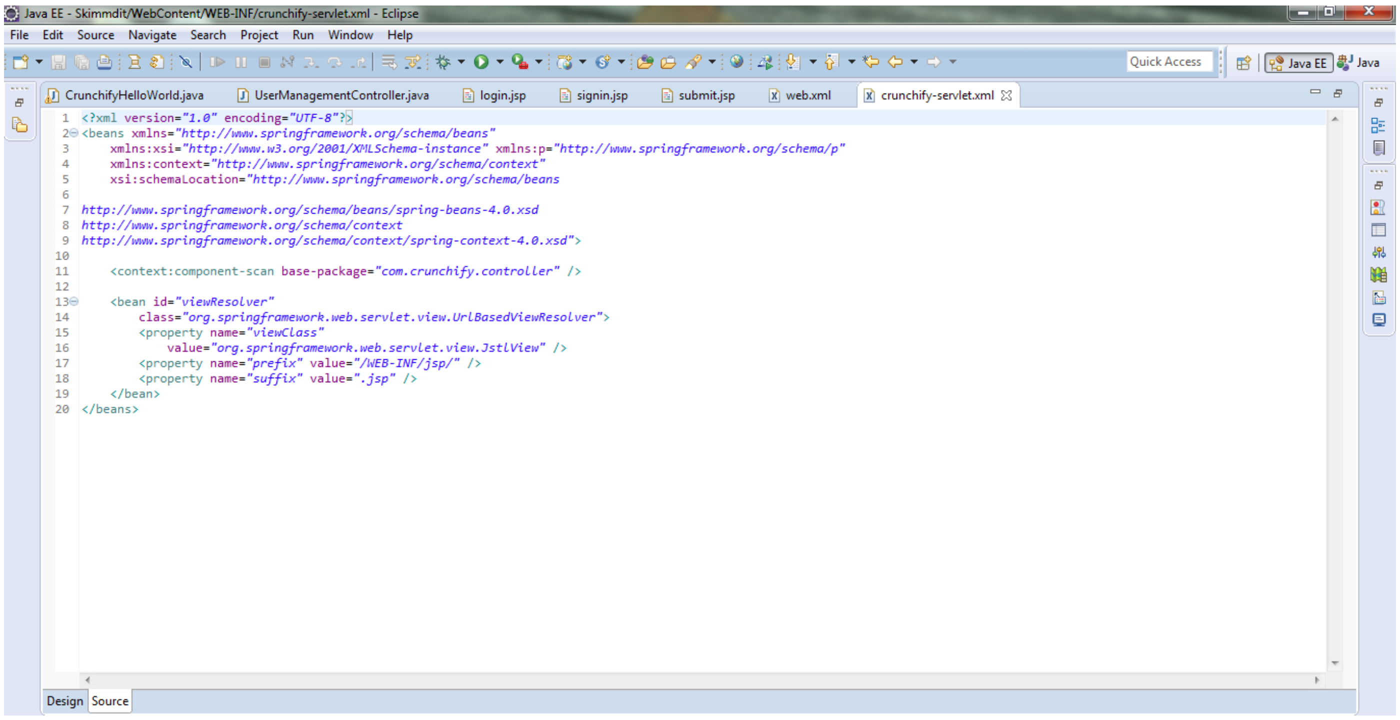
Figure1 : This figure shows a piece of code of our application where in we have configured to use dispatcher servlet. All the servlets used in the previous assignment have been mapped under a single servlet. As shown below, login, signin, logout and so on have been mapped under a single servlet in web.xml file.



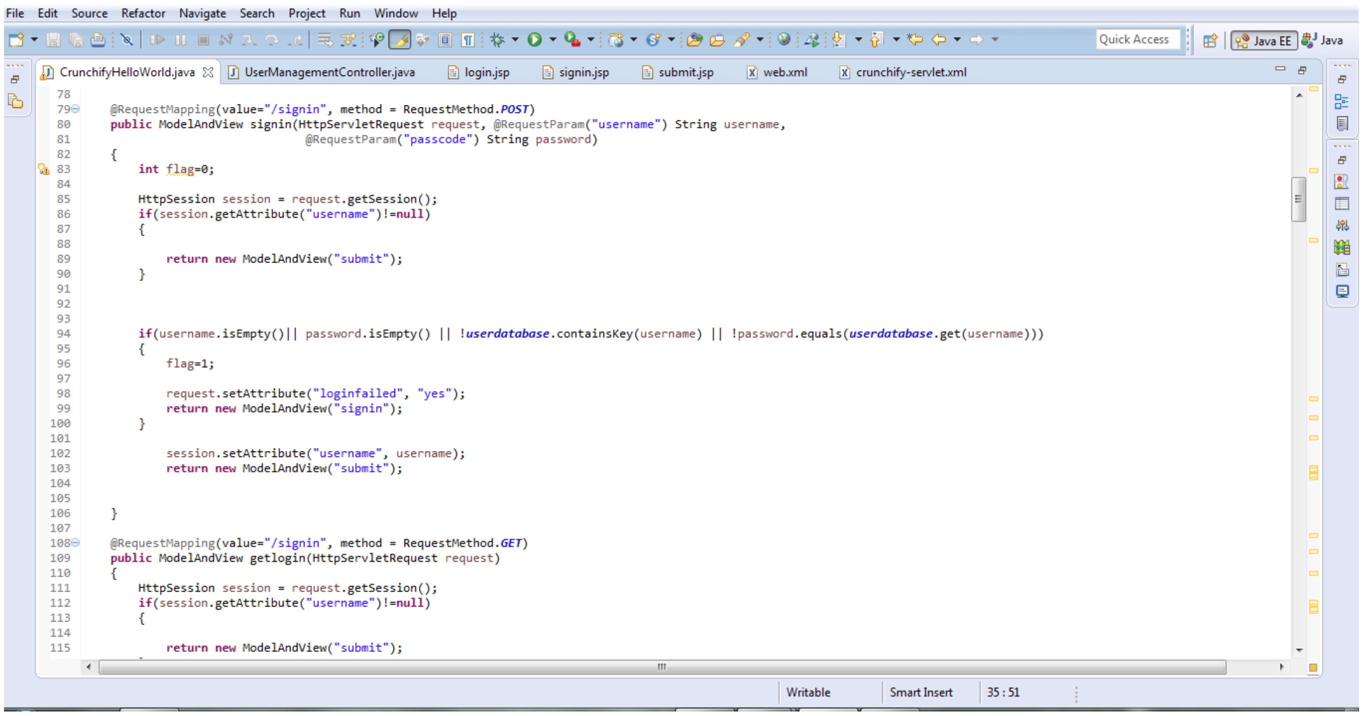
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/javaee" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun
  <display-name>CrunchifySpringMVCTutorial</display-name>
  <welcome-file-list>

    <welcome-file>index.jsp</welcome-file>

  </welcome-file-list>
  <context-param>
<param-name>contextConfigLocation</param-name>
<param-value>/WEB-INF/crunchify-servlet.xml</param-value>
</context-param>
<listener>
<listener-class>org.springframework.web.context.ContextLoaderListener
</listener-class>
</listener>

   <servlet>
        <servlet-name>crunchify</servlet-name>
        <servlet-class>
            org.springframework.web.servlet.DispatcherServlet
        </servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>crunchify</servlet-name>
         <url-pattern>/login</url-pattern>
         <url-pattern>/signin</url-pattern>
          <url-pattern>/logout</url-pattern>
           <url-pattern>/link_handler</url-pattern>
            <url-pattern>/load_dashboard</url-pattern>
            <url-pattern>/vote_handler</url-pattern>
            <url-pattern>/dislike_handler</url-pattern>
    </servlet-mapping>

</web-app>
```

Figure 2: The figure below shows a piece of code, crunchify-servlet.xml, which is our bean configuration file. We have configured our bean to handle all the jsp pages. Every jsp file is called through this bean file as we have configured property value to be .jsp. In the Spring, It recognises the classes by two ways: component scaning and annotations based. We have used component based scanning, It scans the classes within the defined base package.


Java EE - Skimmdit/WebContent/WEB-INF/crunchify-servlet.xml - Eclipse
File Edit Source Navigate Search Project Run Window Help
Quick Access
Java EE
Java
CrunchifyHelloWorld.java
UserManagementController.java
login.jsp
signin.jsp
submit.jsp
web.xml
crunchify-servlet.xml
Design
Source

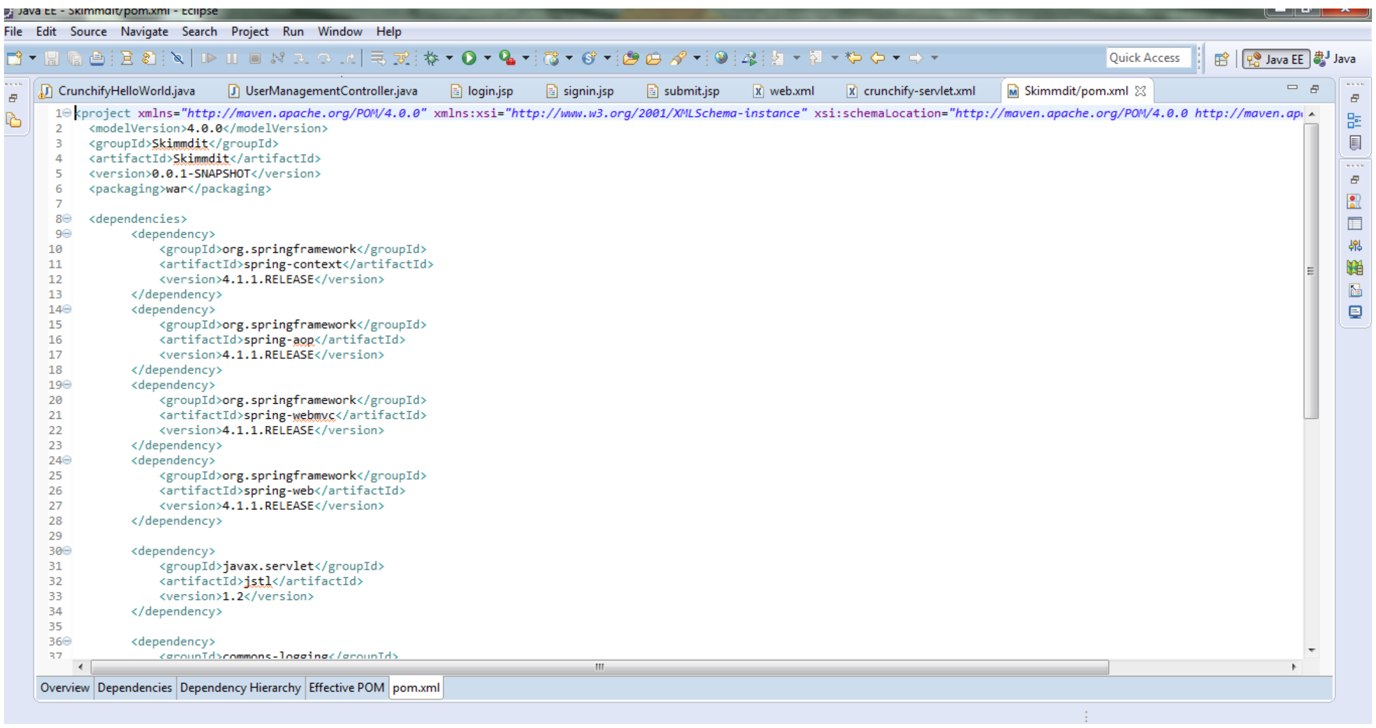
```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:p="http://www.springframework.org/schema/p"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-4.0.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-4.0.xsd">

    <context:component-scan base-package="com.crunchify.controller" />

    <bean id="viewResolver"
        class="org.springframework.web.servlet.view.UrlBasedViewResolver">
        <property name="viewClass"
            value="org.springframework.web.servlet.view.JstlView" />
        <property name="prefix" value="/WEB-INF/jsp/" />
        <property name="suffix" value=".jsp" />
    </bean>
</beans>
```

Figure 3: The /signin is handled through @RequestMapping for both GET and POST requests. It is one of our controller method that also uses the @RequestParam for getting request parameters.



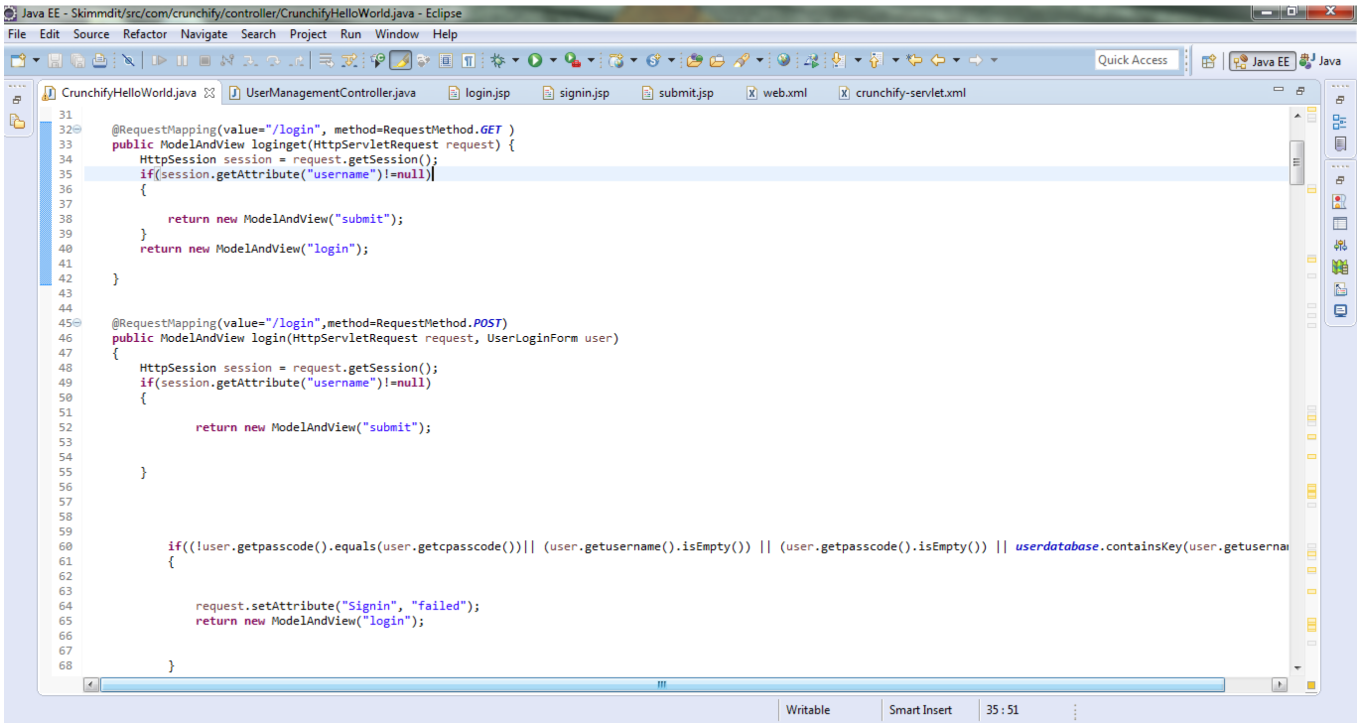
```java

    @RequestMapping(value="/signin", method = RequestMethod.POST)
    public ModelAndView signin(HttpServletRequest request, @RequestParam("username") String username,
                               @RequestParam("passcode") String password)
    {
        int flag=0;

        HttpSession session = request.getSession();
        if(session.getAttribute("username")!=null)
        {

            return new ModelAndView("submit");
        }



        if(username.isEmpty()|| password.isEmpty() || !userdatabase.containsKey(username) || !password.equals(userdatabase.get(username)))
        {
            flag=1;

            request.setAttribute("loginfailed", "yes");
            return new ModelAndView("signin");
        }

            session.setAttribute("username", username);
            return new ModelAndView("submit");


    }

    @RequestMapping(value="/signin", method = RequestMethod.GET)
    public ModelAndView getlogin(HttpServletRequest request)
    {
        HttpSession session = request.getSession();
        if(session.getAttribute("username")!=null)
        {

            return new ModelAndView("submit");
```

Figure 4: Our application's Maven pom.xml handles. Dependencies that we have used for Spring are defined below in screenshots.



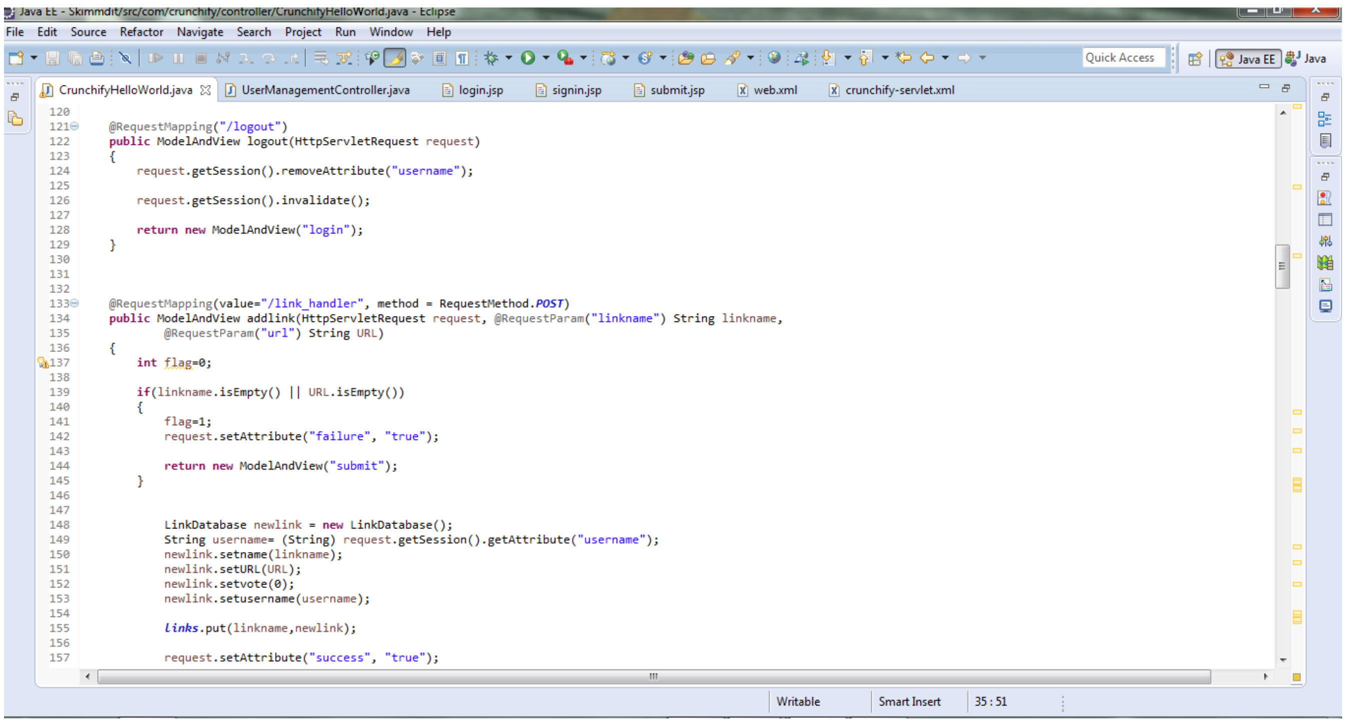
```xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.ap
  <modelVersion>4.0.0</modelVersion>
  <groupId>Skimmdit</groupId>
  <artifactId>Skimmdit</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>war</packaging>

  <dependencies>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>4.1.1.RELEASE</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-aop</artifactId>
            <version>4.1.1.RELEASE</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>
            <version>4.1.1.RELEASE</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-web</artifactId>
            <version>4.1.1.RELEASE</version>
        </dependency>

        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>jstl</artifactId>
            <version>1.2</version>
        </dependency>

        <dependency>
            <groupId>commons-logging</groupId>
```

Figure 5: The figure below shows a piece of code, these are our another controller method. The GET and the POST request is handled under the same request mapping using @Requestmapping annotation.


Java EE - Skimmdit/src/com/crunchify/controller/CrunchifyHelloWorld.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java EE
Java
CrunchifyHelloWorld.java
UserManagementController.java
login.jsp
signin.jsp
submit.jsp
web.xml
crunchify-servlet.xml
Writable
Smart Insert
35 : 51

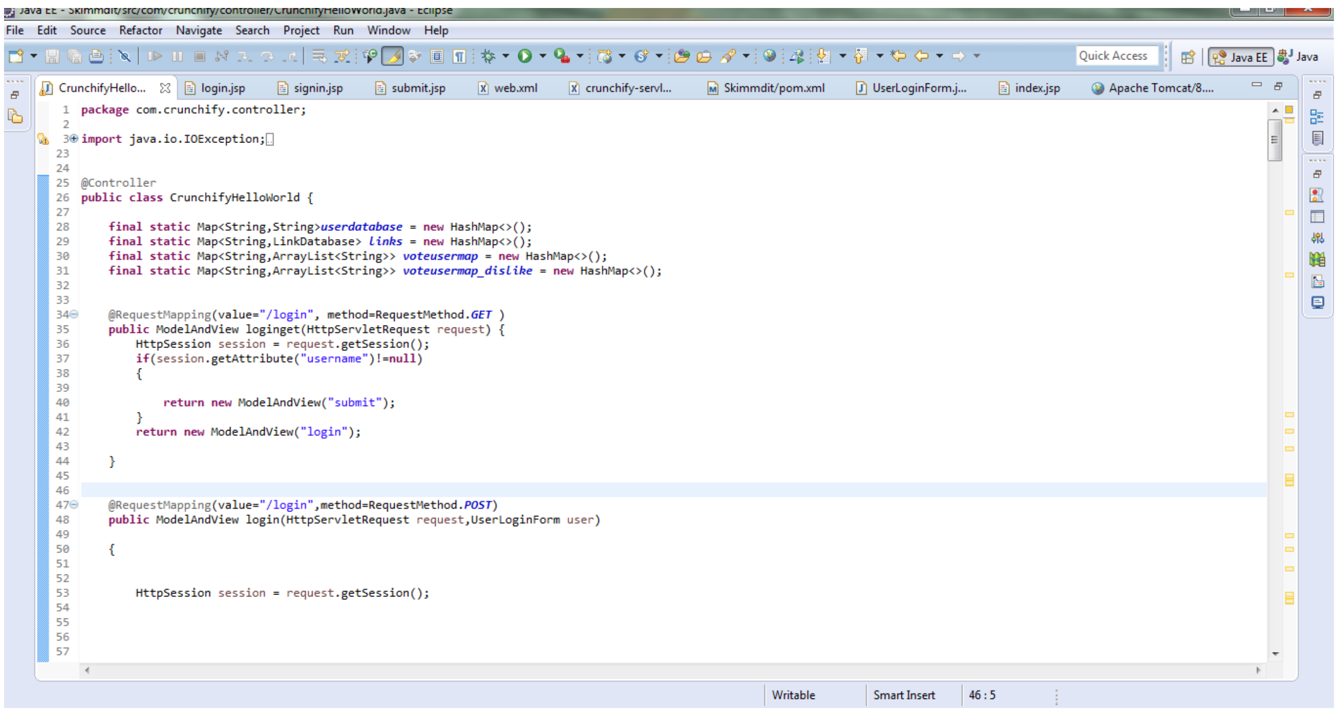
```java

    @RequestMapping(value="/login", method=RequestMethod.GET )
    public ModelAndView loginget(HttpServletRequest request) {
        HttpSession session = request.getSession();
        if(session.getAttribute("username")!=null)
        {

            return new ModelAndView("submit");
        }
        return new ModelAndView("login");

    }


    @RequestMapping(value="/login",method=RequestMethod.POST)
    public ModelAndView login(HttpServletRequest request, UserLoginForm user)
    {
        HttpSession session = request.getSession();
        if(session.getAttribute("username")!=null)
        {

                return new ModelAndView("submit");


        }





            if((!user.getpasscode().equals(user.getcpasscode())|| (user.getusername().isEmpty()) || (user.getpasscode().isEmpty()) || userdatabase.containsKey(user.getusernai
            {


                request.setAttribute("Signin", "failed");
                return new ModelAndView("login");


            }
```

Figure 6: This is our controller method for /logout. The code below shows the usage of annotations like @RequestParam and @RequestMapping applied to method parameters appropriately.


Java EE - Skimmdit/src/com/crunchify/controller/CrunchifyHelloWorld.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java EE
Java
CrunchifyHelloWorld.java
UserManagementController.java
login.jsp
signin.jsp
submit.jsp
web.xml
crunchify-servlet.xml
Writable
Smart Insert
35 : 51

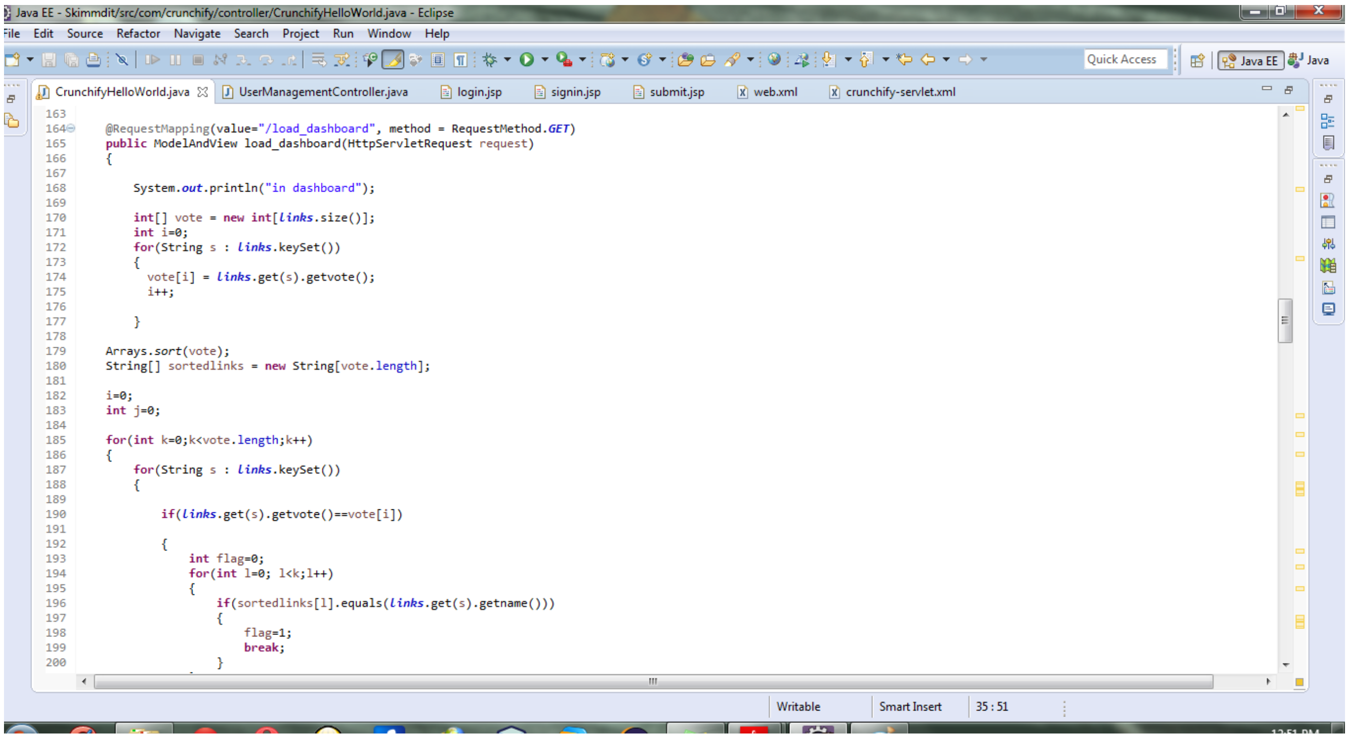
```java

    @RequestMapping("/logout")
    public ModelAndView logout(HttpServletRequest request)
    {
        request.getSession().removeAttribute("username");

        request.getSession().invalidate();

        return new ModelAndView("login");
    }



    @RequestMapping(value="/link_handler", method = RequestMethod.POST)
    public ModelAndView addlink(HttpServletRequest request, @RequestParam("linkname") String linkname,
            @RequestParam("url") String URL)
    {
        int flag=0;

        if(linkname.isEmpty() || URL.isEmpty())
        {
            flag=1;
            request.setAttribute("failure", "true");

            return new ModelAndView("submit");
        }


            LinkDatabase newlink = new LinkDatabase();
            String username= (String) request.getSession().getAttribute("username");
            newlink.setname(linkname);
            newlink.setURL(URL);
            newlink.setvote(0);
            newlink.setusername(username);

            links.put(linkname,newlink);

            request.setAttribute("success", "true");
```

Figure 7: The figure below shows a piece of code where in the servlets are refactored to use spring `@Controller` objects with separate methods for HTTP and URL under get and post.



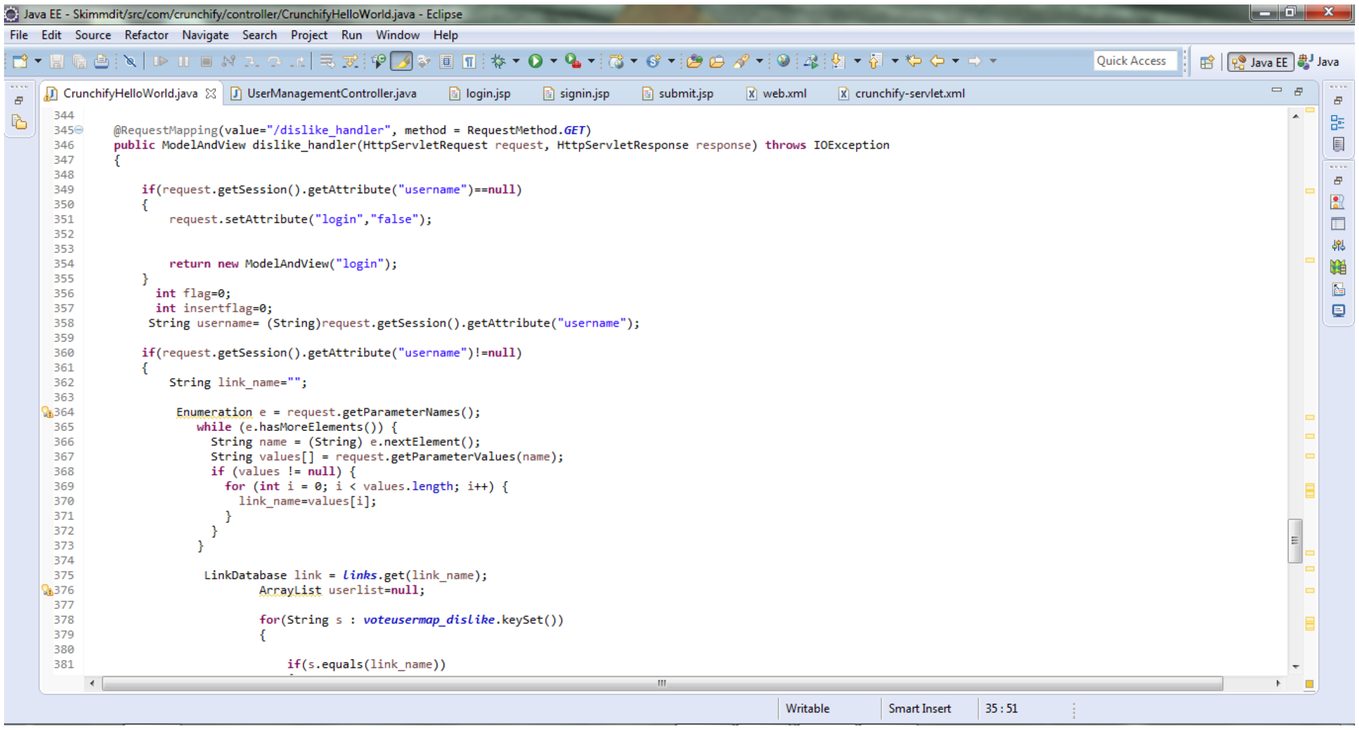
```java
package com.crunchify.controller;

import java.io.IOException;


@Controller
public class CrunchifyHelloWorld {

    final static Map<String,String>userdatabase = new HashMap<>();
    final static Map<String,LinkDatabase> links = new HashMap<>();
    final static Map<String,ArrayList<String>> voteusermap = new HashMap<>();
    final static Map<String,ArrayList<String>> voteusermap_dislike = new HashMap<>();


    @RequestMapping(value="/login", method=RequestMethod.GET )
    public ModelAndView loginget(HttpServletRequest request) {
        HttpSession session = request.getSession();
        if(session.getAttribute("username")!=null)
        {

            return new ModelAndView("submit");
        }
        return new ModelAndView("login");

    }


    @RequestMapping(value="/login",method=RequestMethod.POST)
    public ModelAndView login(HttpServletRequest request,UserLoginForm user)

    {


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

Figure 8: This is our another controller method. The figure shows the piece of code used to handle links and the votes in our main Dashboard. The votes are still in the sorted order(like in assignment1) and also the count of each votes received.



```java

    @RequestMapping(value="/load_dashboard", method = RequestMethod.GET)
    public ModelAndView load_dashboard(HttpServletRequest request)
    {

        System.out.println("in dashboard");

        int[] vote = new int[links.size()];
        int i=0;
        for(String s : links.keySet())
        {
          vote[i] = links.get(s).getvote();
          i++;

        }

    Arrays.sort(vote);
    String[] sortedlinks = new String[vote.length];

    i=0;
    int j=0;

    for(int k=0;k<vote.length;k++)
    {
        for(String s : links.keySet())
        {

            if(links.get(s).getvote()==vote[i])

            {
                int flag=0;
                for(int l=0; l<k;l++)
                {
                    if(sortedlinks[l].equals(links.get(s).getname()))
                    {
                        flag=1;
                        break;
                    }
```

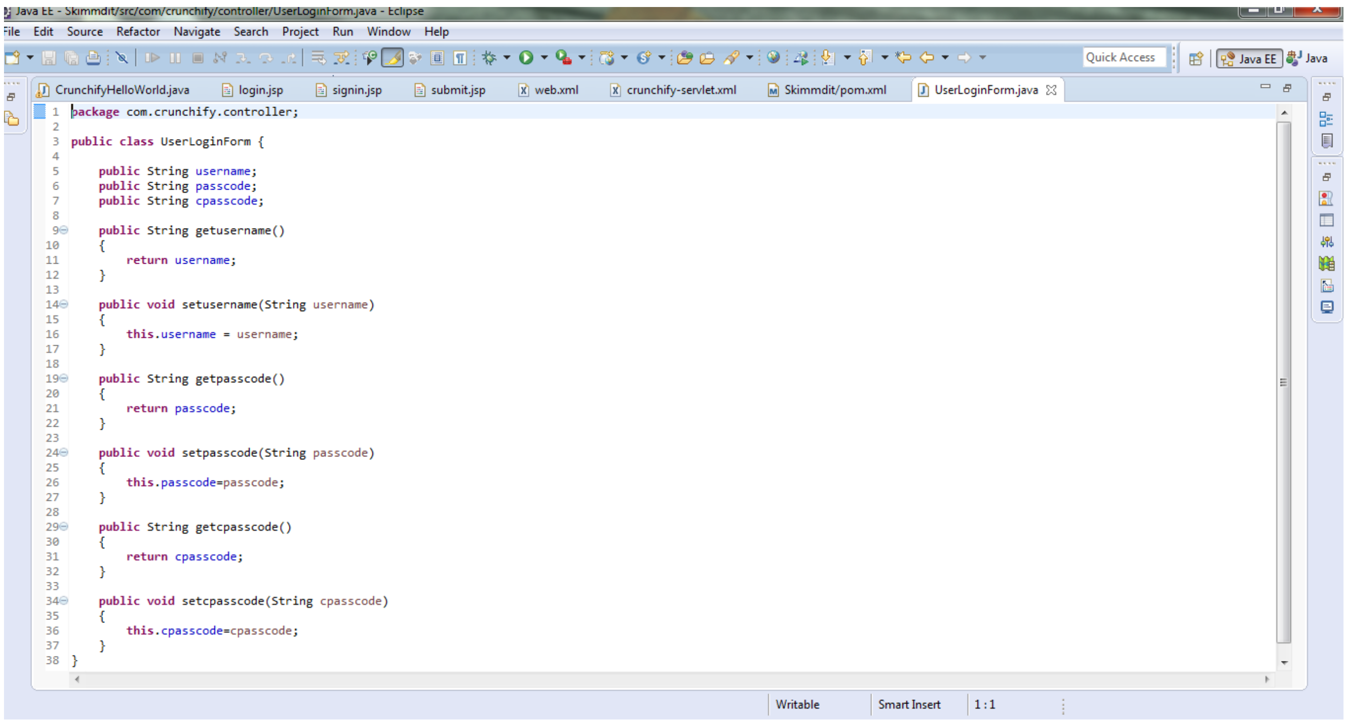
Figure 9: This is our controller method. The figure below shows the /dislike_handler request mapping piece of code, where the HTTP Get method to dislike the votes.



Java EE - Skimmdit/src/com/crunchify/controller/CrunchifyHelloWorld.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java EE
Java
CrunchifyHelloWorld.java
UserManagementController.java
login.jsp
signin.jsp
submit.jsp
web.xml
crunchify-servlet.xml
Writable
Smart Insert
35 : 51

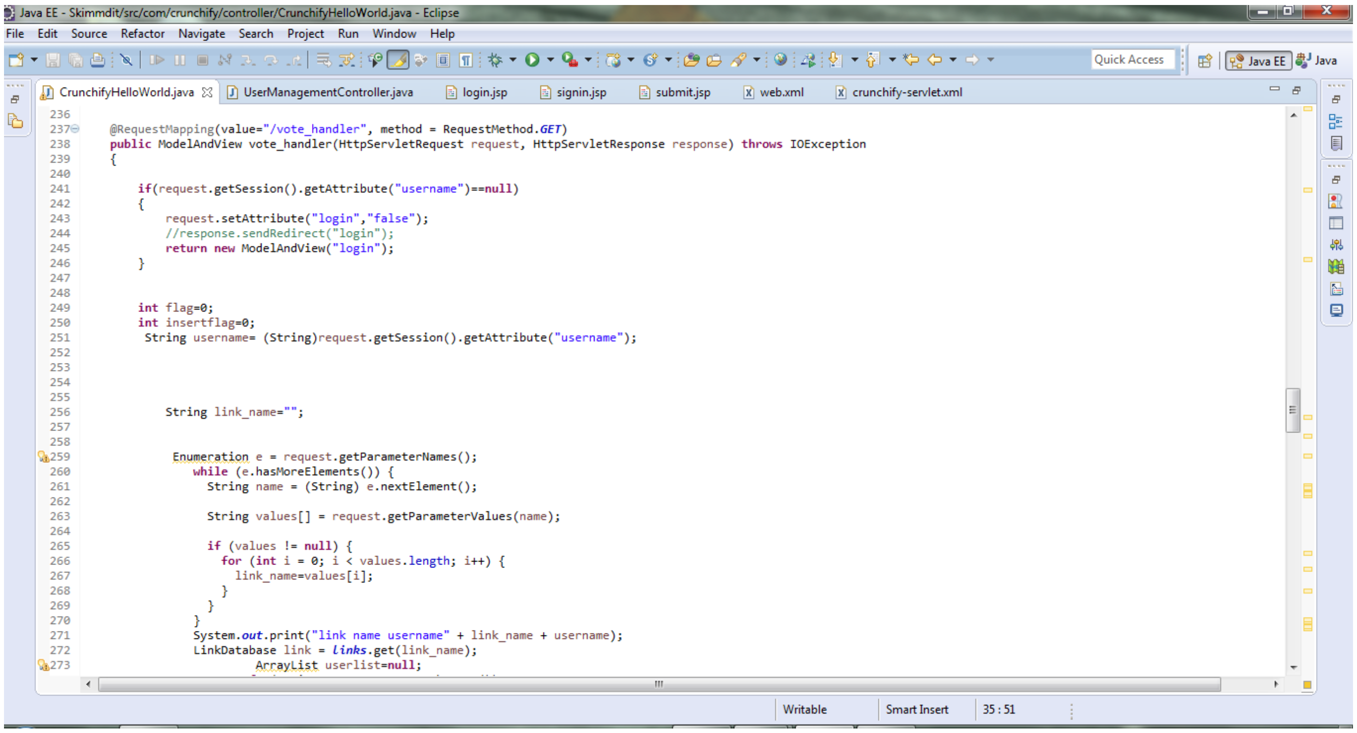
```java

    @RequestMapping(value="/dislike_handler", method = RequestMethod.GET)
    public ModelAndView dislike_handler(HttpServletRequest request, HttpServletResponse response) throws IOException
    {

        if(request.getSession().getAttribute("username")==null)
        {
            request.setAttribute("login","false");


            return new ModelAndView("login");
        }
          int flag=0;
          int insertflag=0;
         String username= (String)request.getSession().getAttribute("username");

        if(request.getSession().getAttribute("username")!=null)
        {
            String link_name="";

             Enumeration e = request.getParameterNames();
                while (e.hasMoreElements()) {
                  String name = (String) e.nextElement();
                  String values[] = request.getParameterValues(name);
                  if (values != null) {
                    for (int i = 0; i < values.length; i++) {
                      link_name=values[i];
                    }
                  }
                }

               LinkDatabase link = links.get(link_name);
                       ArrayList userlist=null;

                       for(String s : voteusermap_dislike.keySet())
                       {

                           if(s.equals(link_name))
```

Figure 10: The below figure shows a piece of code for the UserloginForm, form object will be called by @requestmapping annotation at /login and will be redirected to this page to set and get the username to provide the login credentials.



```java
package com.crunchify.controller;

public class UserLoginForm {

    public String username;
    public String passcode;
    public String cpasscode;

    public String getusername()
    {
        return username;
    }

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

    public String getpasscode()
    {
        return passcode;
    }

    public void setpasscode(String passcode)
    {
        this.passcode=passcode;
    }

    public String getcpasscode()
    {
        return cpasscode;
    }

    public void setcpasscode(String cpasscode)
    {
        this.cpasscode=cpasscode;
    }
}
```

Figure 11: This is our another controller method for request mapping of /vote_handler. This figure shows the Vote Handler mappings, to handle the votes and if the user is not logged in it returns to the login form.


Java EE - Skimmdit/src/com/crunchify/controller/CrunchifyHelloWorld.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java EE
Java
CrunchifyHelloWorld.java
UserManagementController.java
login.jsp
signin.jsp
submit.jsp
web.xml
crunchify-servlet.xml
Writable
Smart Insert
35 : 51

```
@RequestMapping(value="/vote_handler", method = RequestMethod.GET)
public ModelAndView vote_handler(HttpServletRequest request, HttpServletResponse response) throws IOException
{

    if(request.getSession().getAttribute("username")==null)
    {
        request.setAttribute("login","false");
        //response.sendRedirect("login");
        return new ModelAndView("login");
    }


    int flag=0;
    int insertflag=0;
     String username= (String)request.getSession().getAttribute("username");




        String link_name="";


         Enumeration e = request.getParameterNames();
            while (e.hasMoreElements()) {
              String name = (String) e.nextElement();

              String values[] = request.getParameterValues(name);

              if (values != null) {
                for (int i = 0; i < values.length; i++) {
                  link_name=values[i];
                }
              }
            }
            System.out.print("link name username" + link_name + username);
            LinkDatabase link = links.get(link_name);
                     ArrayList userlist=null;
```