Dublin City University

School of Electronics

Assignment 3 Submission

Student Name	**Dhruv Joshi**
Student ID Number	**20216445**
Programme	**Master of Engineering in Electronics and Computer Engineering**
Project Title	**Convert old Web application into a server-side web application**
Module Code	**EE417**
Project Due Date	**22-03-2021**
Lecturer	**Mr Ali Intizar**

Objective

Aim is extend the client-side interface for the Web Application and convert it into server-side web application.

Description of Assignment

- Convert your static web application submitted for Assignment 2 into a server-side application.

- Create a login system for your web application.

    - USER
    - ADMIN
    - EDITOR

- Introduce the following 4 session management techniques in your web application.

    - Http Sessions
    - Cookies
    - URL Rewriting
    - Hidden Form

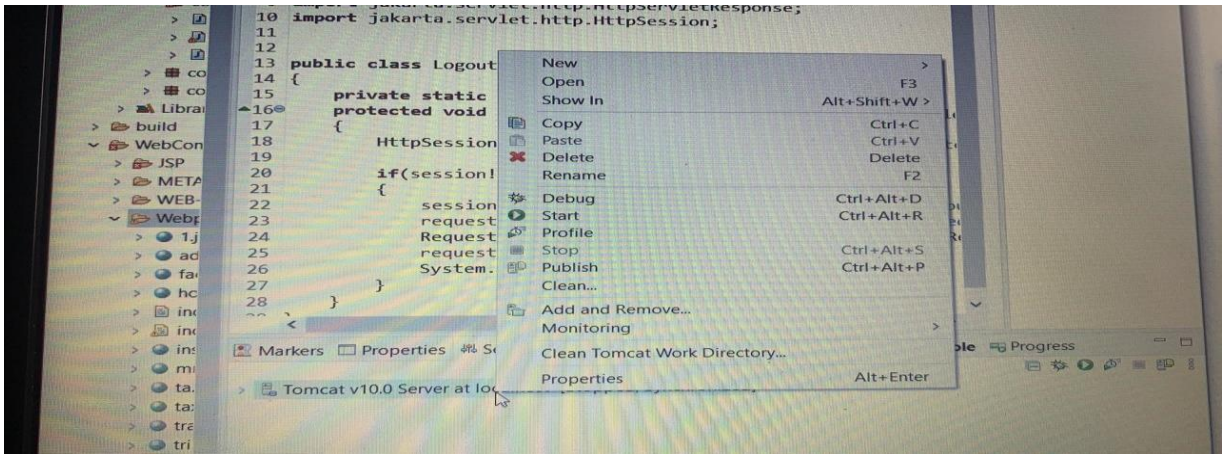
About Assignment 3

In this assignment, we extend the client-side interface for your Web application and convert it into a server-side web application. For this we use IDE eclipse and Apache Tomcat server 10, so that we can we can perform the following task in a smoother way,

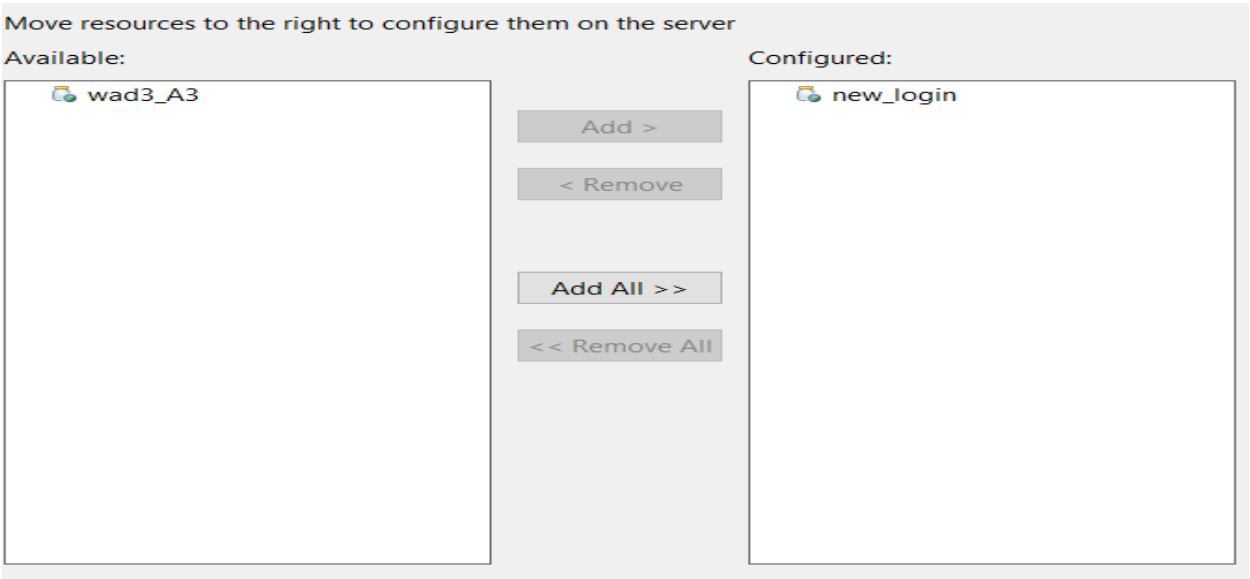
Task 1: Convert your static web application submitted for Assignment 2 into a server-side application.

To perform this task firstly we take our old web application and copy all the sources files (index.html, CSS files and images) to our newly created dynamic project in the eclipse IDE in the Web content folder and then we launch the Tomcat server to perform the following task, for that we start our tomcat server, once the tomcat server is started we perform to run the old project over server and below are results for it.

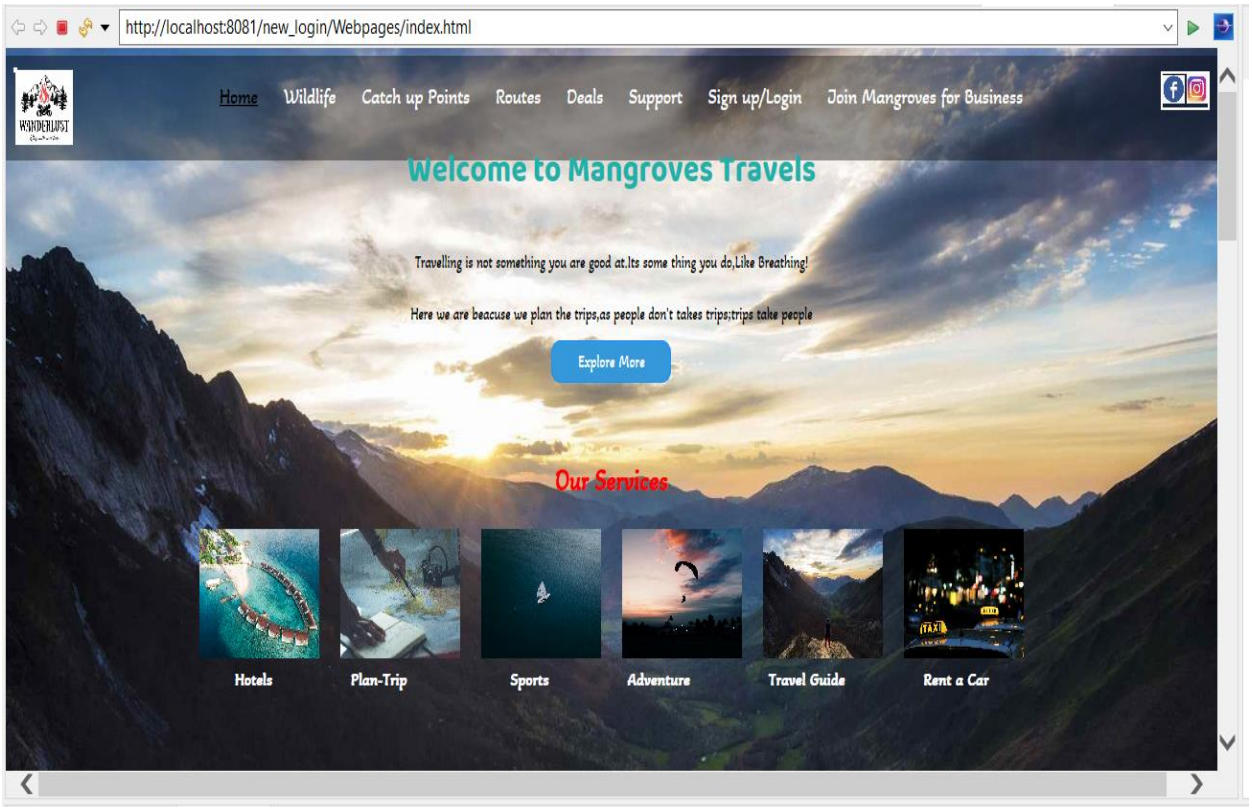
Start the tomcat server,



Configure the File



Final Result,



For the following performed task we receive the below mentioned address,

http://localhost:8081/new_login/Webpages/index.html

Task 2: Create a login system for your web application.

Here we plan to create a login page that is only responsive once the username and password is being placed into it, and for login page we create three parts for it User, Admin and Editor.

Here we created three JSP pages based on 3 authorized roles **USER, ADMIN and EDITOR.**

The landing page for our application is a login page and the code is written in the login.jsp

```
<table align="center">
<tr>
<td>Username</td>
<td><input type="text" name="username" /></td>
</tr>
<tr>
<td>Password</td>
<td><input type="password" name="password" /></td>
</tr>
<tr>
```

This page contains the username and password and the login button and also the invalid credentials message. The password field is of type password so that the password is hidden.

Explanation of Admin.jsp,

This page contains the logout link and the username and the title for the page.

In this page we are checking whether the user session has admin role or not. If the session doesn't have admin role then we forward the request to the login page. **Username: Dhruv |Password: abc**

Once the logout is clicked then we hit the Logout servlet and clear the session.

```
<title>Admin Page</title>
</head>
<% //In case, if Admin session is not set, redirect to Login page
if((request.getSession(false).getAttribute("Admin")== null) )
{
%>
<jsp:forward page="/JSP/Login.jsp"></jsp:forward>
<%} %>
<body>
<center><h2>Admin's Home</h2></center>

Welcome <%=request.getAttribute("userName") %>

<div style="text-align: right"><a href="<%=request.getContextPath()%>/LogoutServlet">Logout</a></div>
```

Explanation of User.jsp,

This page contains the logout link and the username and the title for the page.

In this page we are checking whether the user session has User role or not. If the session doesn't have User role then we forward the request to the login page. **Username: DCU |Password: abc**

Once the logout is clicked then we hit the Logout servlet and clear the session.

```
<title>User Page</title>
</head>
<% //In case, if User session is not set, redirect to Login page.
if((request.getSession(false).getAttribute("User")== null) )
{
%>
<jsp:forward page="/JSP/Login.jsp"></jsp:forward>
<%} %>
<body>
    <center><h2>User's Home</h2></center>
    Welcome <%=request.getAttribute("userName") %>

    <div style="text-align: right"><a href="<%=request.getContextPath()%>/LogoutServlet">Logout</a></div>
```

Explanation of Editor.jsp,

This page contains the logout link and the username and the title for the page.

In this page we are checking whether the user session has Editor Role or not. If the session doesn't have Editor Role then we forward the request to the login page. **Username: Professor |Password: abc**

Once the logout is clicked then we hit the Logout servlet and clear the session.

```
<title>Editor Page</title>
</head>
<% //In case, if Editor session is not set, redirect to Login page
if((request.getSession(false).getAttribute("Editor")== null) )
{
%>
<jsp:forward page="/JSP/Login.jsp"></jsp:forward>
<%} %>
<body>
<center><h2>Editor's Home</h2></center>

Welcome <%=request.getAttribute("userName") %>

<div style="text-align: right"><a href="<%=request.getContextPath()%>/LogoutServlet">Logout</a></div>
```

Explanation over the final call,

At the final step when we run the final case over the servers all the feature properly works and the User Login i.e. **User: DCU ,Password: abc** directs us to the requested page i.e. assignment 2 web page but there the CSS styling was not responsive also we have tried the below functionalities to obtain the same but still it won't work ,refer the below snippet

http://localhost:8081/new_login/LoginServlet

```
        request.getRequestDispatcher("/JSP/User.jsp").forward(request, response);
          request.getRequestDispatcher("/Webpages/index.html").forward(request, response);
    }
    else
    {
```

Task 3

Explanation about the Java Code

Our code has three layers and its follows the MVC (Model, view, controller) architecture

- **LoginBean:**
  Thus class is a model for our application and the structure is for the user. It has three field and getters and setters

```
private String userName;
private String password;
private String role;
```

Controllers:
We have three controllers Loginservlet, login servlet with cookie, logout servlet.

- **Login servlet**
  This servlet implements the http: session mechanism and it is hit when the URL path is `/LoginServlet`

```
<servlet-mapping>
    <servlet-name>LoginServlet</servlet-name>
    <url-pattern>/LoginServlet</url-pattern>
</servlet-mapping>
```

Once the request comes to servlet i.e. when you click the submit button

Once you click the login button then the doPost method is getting called and we capture the username and password entered by the user form the `HttpServletRequest` parameter and we pass this value to the LoginDao class to authenticate and return the role or invalidate the user message.

If the user is authenticated then we assign role in the http session and forward the request to load jsp pages based on the role.

The code snippet below shows the session mechanism implemented in this app. Here we have set 10 minutes session inactivity timeout.

```
if(userValidate.equals("Admin_Role"))
{
    System.out.println("Admin's Home");

    HttpSession session = request.getSession(); //Creating a session
    session.setMaxInactiveInterval(10*60);
    session.setAttribute("Admin", userName); //setting session attribute
    request.setAttribute("userName", userName);

    request.getRequestDispatcher("/JSP/Admin.jsp").forward(request, response);
}
```

- **LoginServletwithCookies**

The code snippet below implements the cookies mechanism.

Here we have set maximum cookies expiry time out as 35min.

```
if(userValidate.equals("Admin_Role"))
{
    System.out.println("Admin's Home");

    setCookie(userName, response);
    request.getRequestDispatcher("/JSP/Admin.jsp").forward(request, response);
}
```

```
public HttpServletResponse setCookie(String username, HttpServletResponse response) {
    Cookie userCookie = new Cookie("user", username);
    userCookie.setMaxAge(35*60);
    response.addCookie(userCookie);
    return response;
}
```

- **LogutServlet**

In this file we invalidate the session and we forward the request to load the login screen with message with a logout successful message.

```
if(session!=null) //If session is not null
{
    session.invalidate(); //removes all session attributes bound to the session
    request.setAttribute("errMessage", "You have logged out successfully");
    RequestDispatcher requestDispatcher = request.getRequestDispatcher("/JSP/Login.jsp");
    requestDispatcher.forward(request, response);
    System.out.println("Logged out");
}
```

- **LoginDao**

In this file we have hardcoded three users and stored them in array list , in order to get a perfect validation we have written a method authenticateUser that scans the array list and if it finds the user then it return the role otherwise it will return a invalid message.

```
ArrayList<LoginBean> users = new ArrayList<LoginBean>();
users.add(new LoginBean("Dhruv", "abc", "Admin"));
users.add(new LoginBean("Proffesor", "abc", "Editor"));
users.add(new LoginBean("DCU", "abc", "User"));
String userName = loginBean.getUserName();
String password = loginBean.getPassword();


String userNameDB = "";
String passwordDB = "";
String roleDB = "";

Iterator<LoginBean> itr = users.listIterator();
while(itr.hasNext())
{
    LoginBean userCurrent = itr.next();
    userNameDB = userCurrent.getUserName();
    passwordDB = userCurrent.getPassword();
    roleDB = userCurrent.getRole();

    if(userName.equals(userNameDB) && password.equals(passwordDB) && roleDB.equals("Admin"))
        return "Admin_Role";
    else if(userName.equals(userNameDB) && password.equals(passwordDB) && roleDB.equals("Editor"))
        return "Editor_Role";
    else if(userName.equals(userNameDB) && password.equals(passwordDB) && roleDB.equals("User"))
        return "User_Role";
}
return "Invalid user credentials";
```

Cookies

Cookies can be described as a small text files placed on a user machine (computer) which are commonly said to collect personal data when they are allowed .Most website operators place cookies on the browser or hard dive of their computer. Cookies are able to collects the information about the use of a website or enable the website to recognise the user as an existing customer when they return to website at a later date or time.

There are 5 types of cookies

- **Session cookies**
- **Permanent cookies**
- **Third party cookies**
- **Flash cookies**
- **Zombie cookies**

The reason behind to opt the cooking mechanism as the suitable one,

Because it usually helps the user to make the current web experience faster, convenient and personalised. For example we select a language to view a website the first time you visit it. When we visit the website again it will save our preference. Also in our case its helps the user to save their particular credentials for a good span of time so that they can check and visit the following part and check the website and remain in the operation until the time is not completed.

Also the reason for choosing the cookies is its laws

- Tell users' cookies are there and what cookies are used for
- Explain what the cookies are doing and why, and
- Get the user consent to store the cookies on their device.