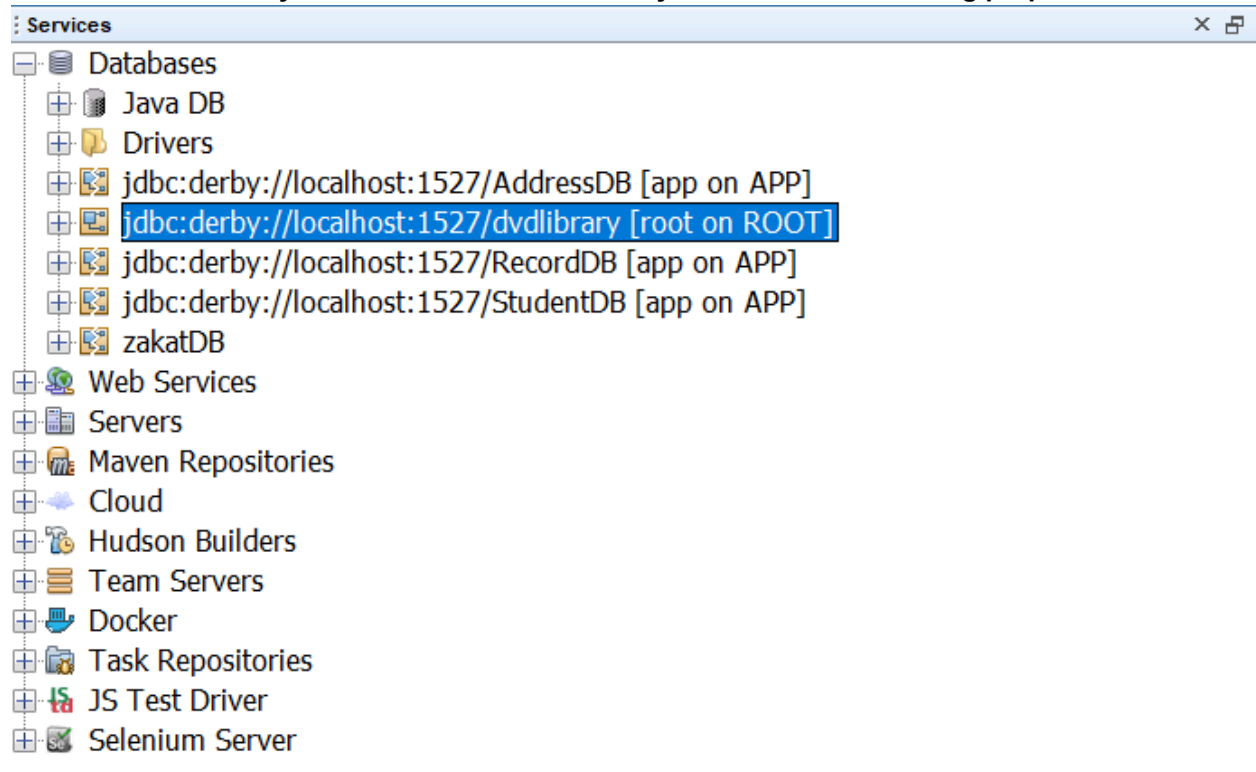
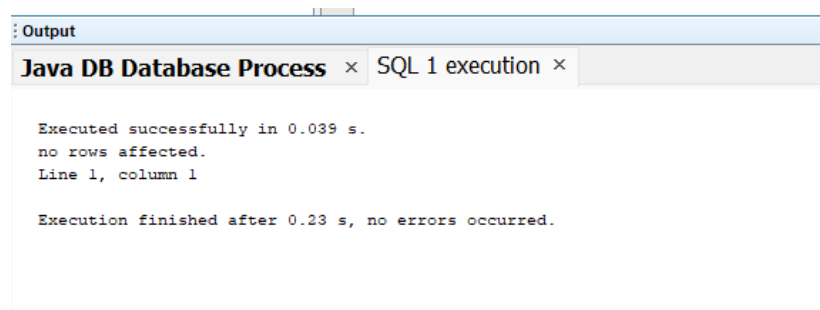
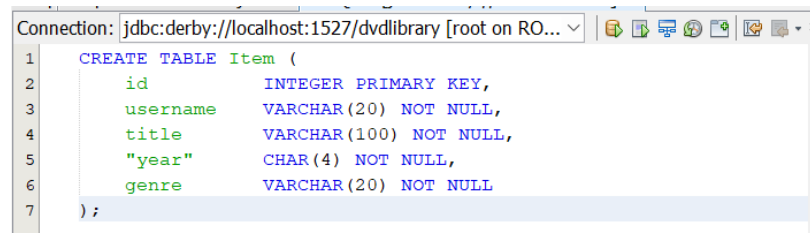


Post LAB 9: Integrating JSP with a database

Task 1: Start the Derby database. Create a dvdlibrary database with following properties



Task 2: Create a table name ITEM by executing the following SQL command.



Task 3: Populate the table with values through SQL command.

```
INSERT INTO Item VALUES(1, 'jack', 'The Godfather', '1972', 'Drama');
INSERT INTO Item VALUES(2, 'jack', 'The Shawshank Redemption', '1994', 'Drama');
INSERT INTO Item VALUES(3, 'jack', 'The Godfather: Part II', '1974', 'Drama');
INSERT INTO Item VALUES(4, 'jack', 'The Lord of the Rings: The Return of The King', '2003', 'Fantasy');
INSERT INTO Item VALUES(5, 'jack', 'The Lord of the Rings: The Two Towers', '2002', 'Fantasy');
INSERT INTO Item VALUES(6, 'jack', 'Shindler's List', '1998', 'Drama');
INSERT INTO Item VALUES(7, 'jack', 'Shichinin no samurai', '1954', 'Action');
INSERT INTO Item VALUES(8, 'jack', 'Casablanca', '1942', 'Drama');
INSERT INTO Item VALUES(9, 'jack', 'The Lord of the Rings: The Fellowship of The Ring', '2001', 'Fantasy');
```

Output ×

Java DB Database Process × SQL 1 execution ×

```
Executed successfully in 0.002 s.
1 row affected.
Line 7, column 1
```

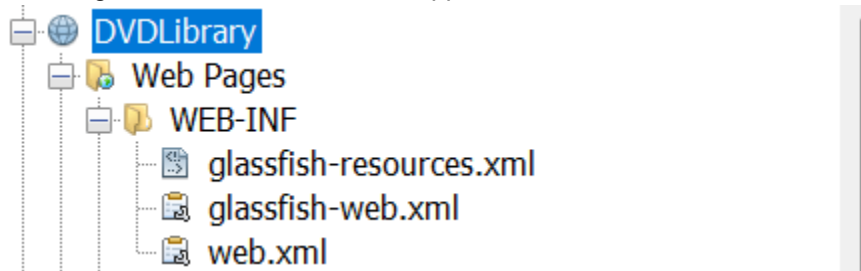
```
Executed successfully in 0.002 s.
1 row affected.
Line 8, column 1
```


```
Executed successfully in 0.002 s.
1 row affected.
Line 9, column 1
```

```
Execution finished after 0.058 s, no errors occurred.
```

Task 4: Perform the following steps:

1. Configure the Data source in the deployment descriptor
2. Configure the data source in the application server



 Add Resource Reference ✕

Resource Name:

jdbc/myDatasource

Resource Type:

javax.sql.DataSource ▾

Authentication:

Container ▾

Sharing Scope:

Shareable ▾

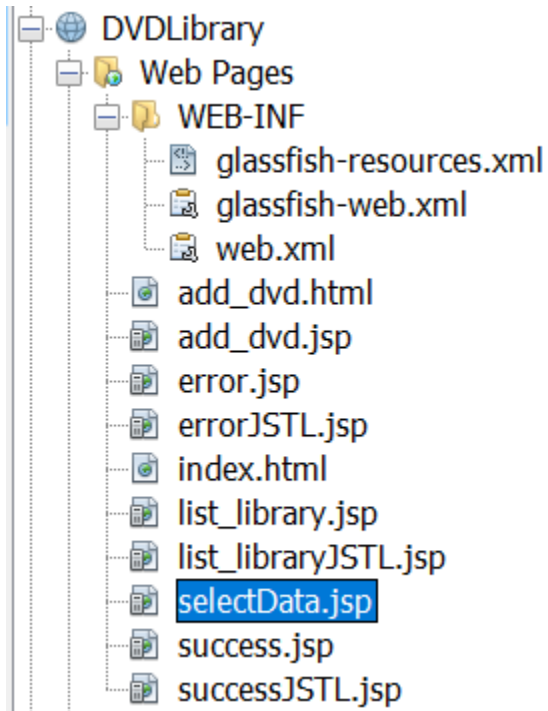
Description:

OK

Cancel

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE glassfish-web-app PUBLIC "-//GlassFish.org//DTD GlassFish Application Server 3.1 Servlet 3.0//EN" "http://glassfish.org/dtds/glassfish-web-app_3_0-1.dtd">
3  <glassfish-web-app error-url="">
4    <context-root>/DBExample</context-root>
5    <class-loader delegate="true"/>
6    <jsp-config>
7      <property name="keepgenerated" value="true">
8        <description>Keep a copy of the generated servlet class' java code.</description>
9      </property>
10   </jsp-config>
11   <resource-ref>
12     <res-ref-name>jdbc/myDatasource</res-ref-name>
13     <jndi-name>jdbc/myDatasource</jndi-name>
14   </resource-ref>
15 </glassfish-web-app>
16
```

3. Create a new JSP file as selectData.jsp. Write JSP/ JSTL code to display the data from database as follow:

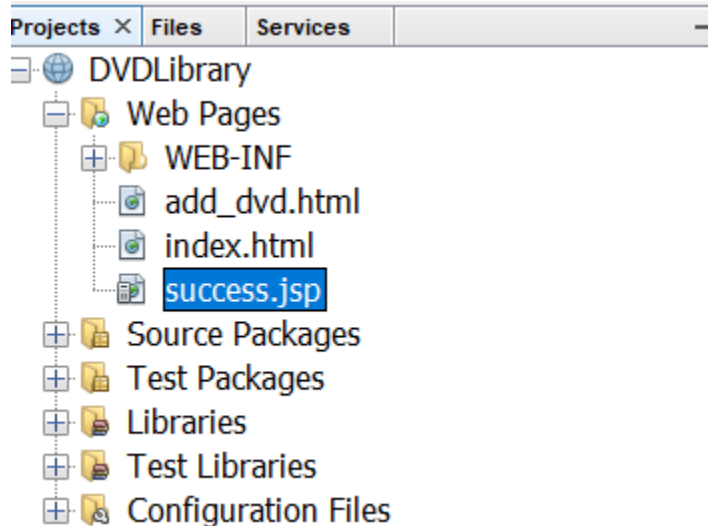


LAB 7: JSP - CSC584

1. Exercise 1: Converting the Add DVD Success View Component to a JSP Page

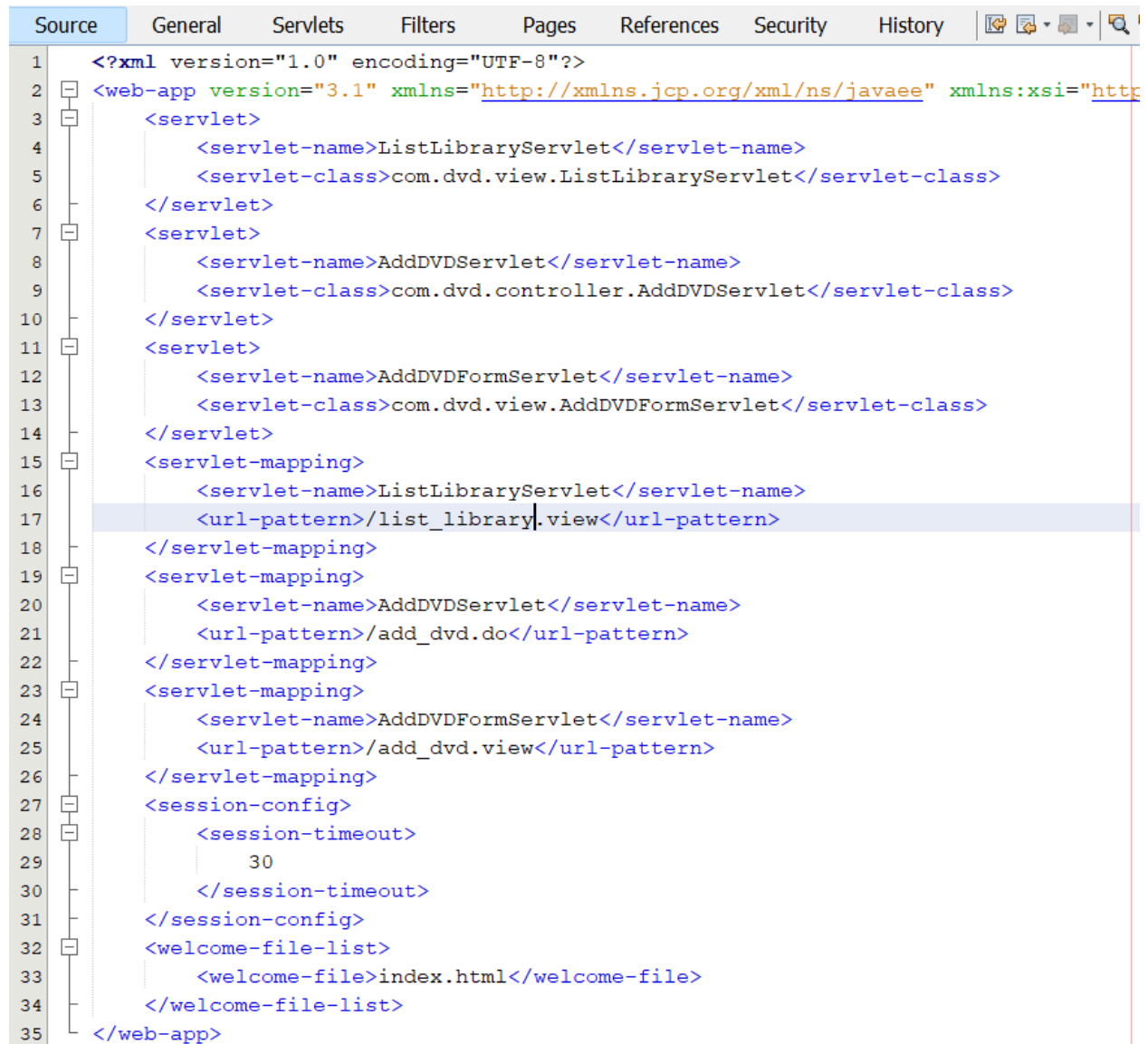
Task 1: Convert the Success Servlet

1. Create JSP file with the following characteristics:



```
Source History
1  <%--
2      Document    : success
3      Created on  : Nov 19, 2024, 5:12:11 PM
4      Author     : nasru
5  --%>
6
7  <%@page contentType="text/html" pageEncoding="UTF-8"%>
8  <!DOCTYPE html>
9  <jsp:useBean id="dvdItem" scope="request" type="com.dvd.model.DVDItem" />
10
11 <html>
12 <head>
13     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
14     <title>DVD Library Application: Add DVD Success</title>
15 </head>
16 <body>
17     <h1>Add DVD Success (JSP)</h1>
18
19     You have add the following DVD: <br/>
20     Title: <jsp:getProperty name="dvdItem" property="title" /><br>
21     Year Release: <jsp:getProperty name="dvdItem" property="year" /><br>
22     Genre of film: <jsp:getProperty name="dvdItem" property="genre" /><br>
23     <br><a href='index.html'>Home</a>
24 </body>
25 </html>
26
```

Task 2: Remove the Old Servlet Code (optional)



```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xmlns:xsi="http://xmlns.jcp.org/xml/ns/javaee/xsi" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd">
3      <servlet>
4          <servlet-name>ListLibraryServlet</servlet-name>
5          <servlet-class>com.dvd.view.ListLibraryServlet</servlet-class>
6      </servlet>
7      <servlet>
8          <servlet-name>AddDVDServlet</servlet-name>
9          <servlet-class>com.dvd.controller.AddDVDServlet</servlet-class>
10     </servlet>
11     <servlet>
12         <servlet-name>AddDVDFormServlet</servlet-name>
13         <servlet-class>com.dvd.view.AddDVDFormServlet</servlet-class>
14     </servlet>
15     <servlet-mapping>
16         <servlet-name>ListLibraryServlet</servlet-name>
17         <url-pattern>/list_library.view</url-pattern>
18     </servlet-mapping>
19     <servlet-mapping>
20         <servlet-name>AddDVDServlet</servlet-name>
21         <url-pattern>/add_dvd.do</url-pattern>
22     </servlet-mapping>
23     <servlet-mapping>
24         <servlet-name>AddDVDFormServlet</servlet-name>
25         <url-pattern>/add_dvd.view</url-pattern>
26     </servlet-mapping>
27     <session-config>
28         <session-timeout>
29             30
30         </session-timeout>
31     </session-config>
32     <welcome-file-list>
33         <welcome-file>index.html</welcome-file>
34     </welcome-file-list>
35 </web-app>
```

Task 3: Deploy the Web Application

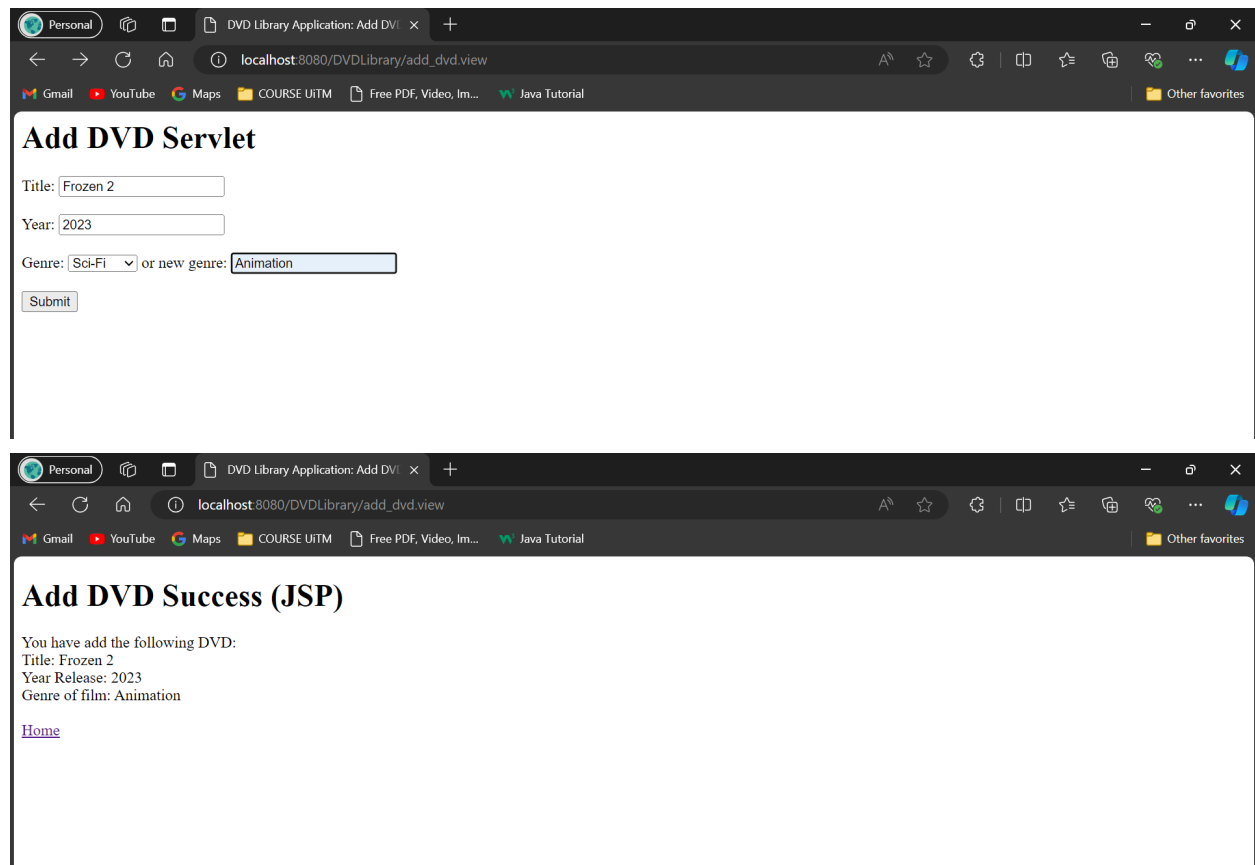


The image shows the NetBeans IDE's output window with the 'Output' tab selected. It displays the execution of an Ant build script for the DVDLibrary project. The build process includes steps for initializing the project, downloading dependencies (module-jar, ear-jar, jar), including libraries in the archive and manifest, compiling the code, and compiling JSPs. The final step is an in-place deployment to the web directory. The output concludes with 'BUILD SUCCESSFUL (total time: 0 seconds)'.

```
ant -f C:\Users\nasru\OneDrive\Documents\NetBeansProjects\DVDLibrary -Dnb.internal.action.name=redeploy -Ddirec
init:
deps-module-jar:
deps-ear-jar:
deps-jar:
library-inclusion-in-archive:
library-inclusion-in-manifest:
compile:
compile-jsp:
In-place deployment at C:\Users\nasru\OneDrive\Documents\NetBeansProjects\DVDLibrary\build\web
run-deploy:
BUILD SUCCESSFUL (total time: 0 seconds)
```

Task 4: Test the Web Application

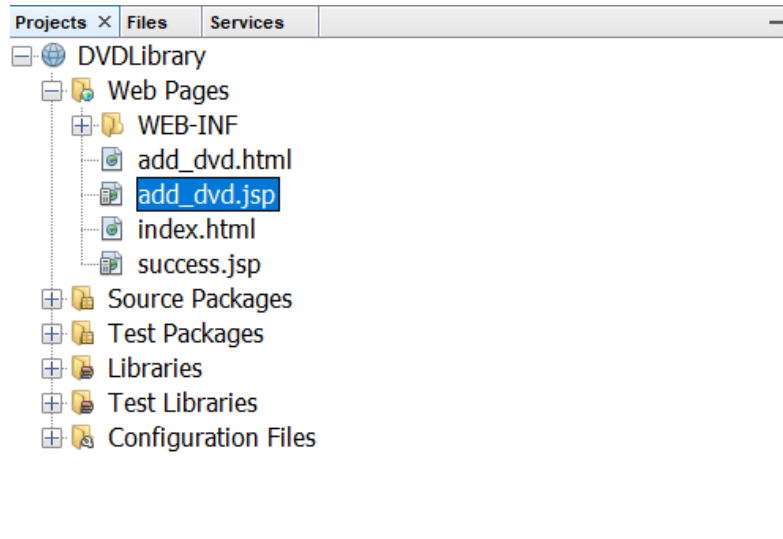
1. Refresh the DVDLibrary web application home page. Test the Add DVD form page.



Exercise 2: Converting the Add DVD Form Component to a JSP Page

Task 1: Convert the AddDVDFormServlet Servlet

1.



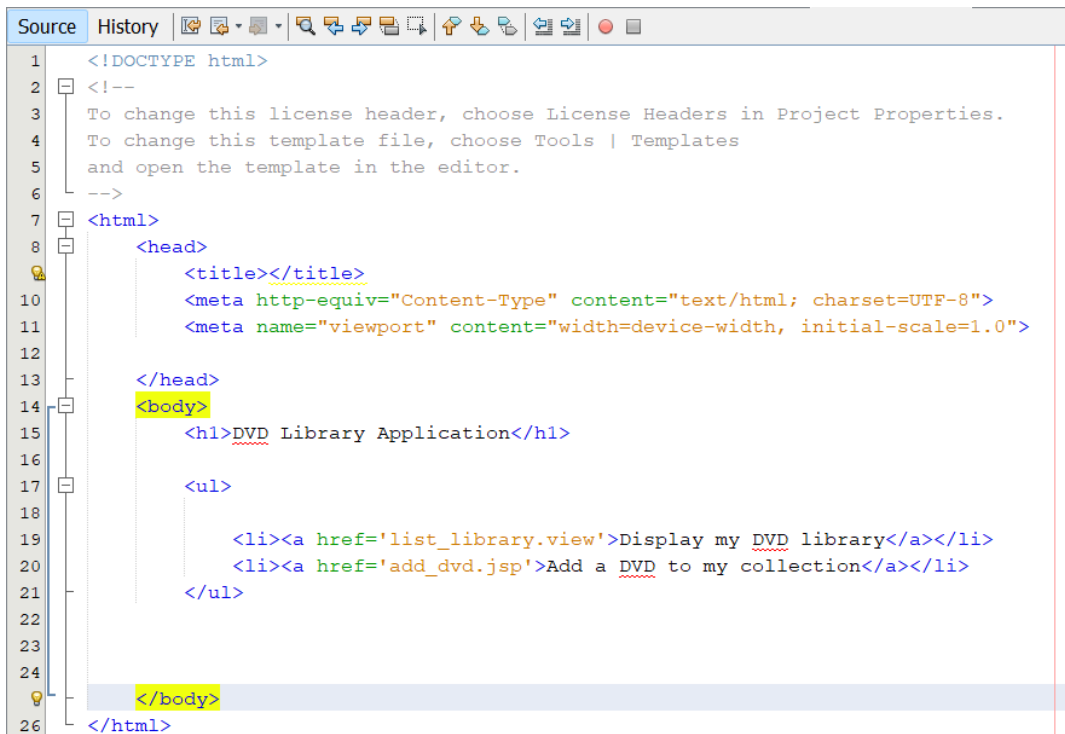
2. + 3

```
Source History
Document : add_dvd
Created on : Nov 19, 2024, 6:59:22 PM
Author : nasru

<!--
-->

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<title>DVD Library Application: Add DVD Form</title>
</head>
<body>
<h1>Add DVD</h1>
<form action='add_dvd.do' method='POST'>
Title: <input type='text' name='title'> <br/> <br/>
<%
    java.util.Calendar calendar = java.util.Calendar.getInstance();
    int currYear = calendar.get(java.util.Calendar.YEAR);
%>
Year: <input type='text' name='year' value="<%= currYear %>"> <br/> <br/>
Genre <select name='genre'>
    <option value='Sci-Fi'>Sci-Fi</option>
    <option value='Drama'>Drama</option>
    <option value='Comedy'>Comedy</option>
</select>
or new genre: <input type='text' name='newGenre'> <br/> <br/>
<input type='submit'>
</form>
</body>
</html>
```

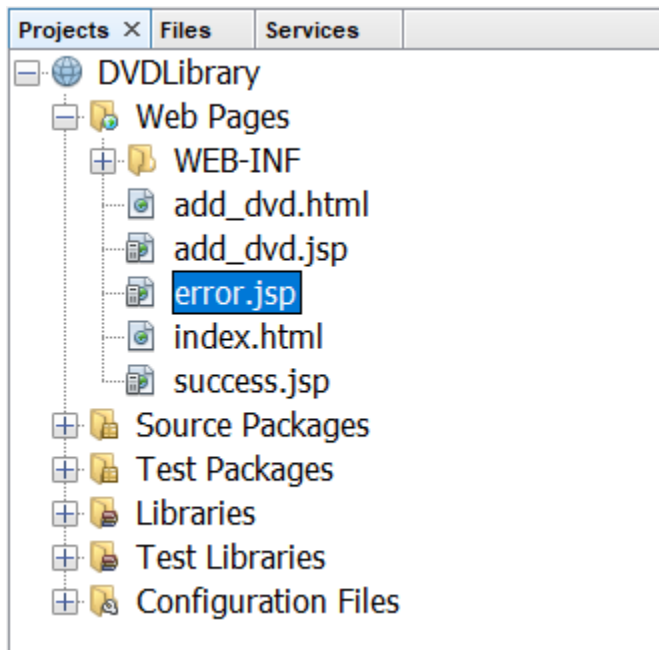

4.



```
1 <!DOCTYPE html>
2 <!--
3 To change this license header, choose License Headers in Project Properties.
4 To change this template file, choose Tools | Templates
5 and open the template in the editor.
6 -->
7 <html>
8 <head>
9 <title></title>
10 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
11 <meta name="viewport" content="width=device-width, initial-scale=1.0">
12
13 </head>
14 <body>
15 <h1>DVD Library Application</h1>
16
17 <ul>
18
19 <li><a href='list_library.view'>Display my DVD library</a></li>
20 <li><a href='add_dvd.jsp'>Add a DVD to my collection</a></li>
21 </ul>
22
23
24 </body>
25
26 </html>
```

Task 2: Create the JSP error page

1. Create JSP file with the following characteristics:

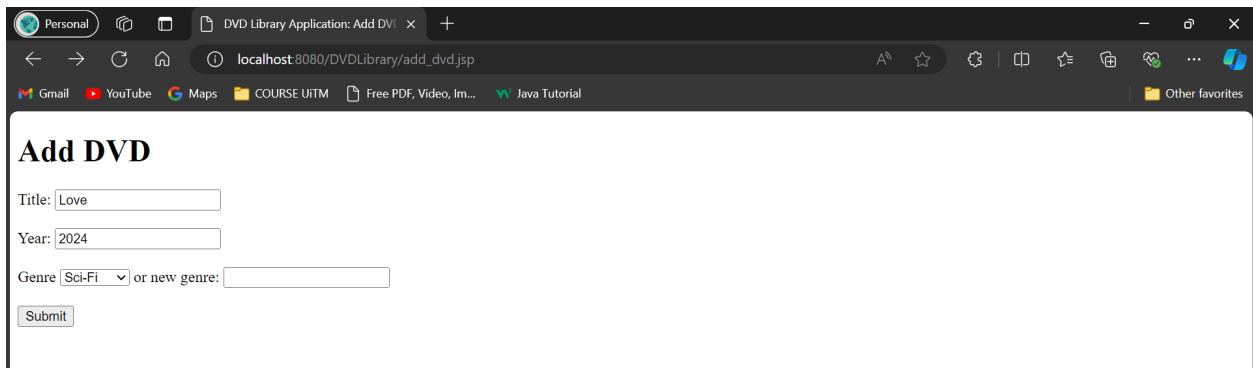


2. + 3

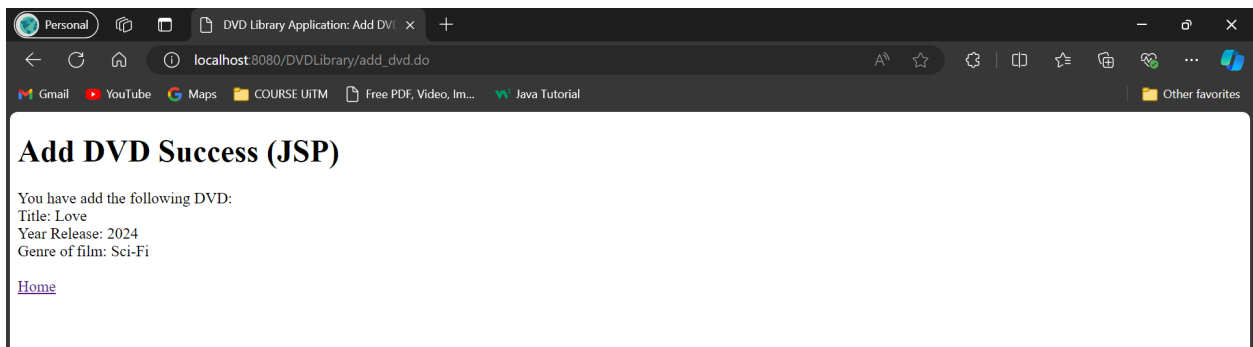
```
46     }
47     if(!errorMsgs.isEmpty()){
48         request.setAttribute("errorMsgs", errorMsgs);
49         RequestDispatcher view = request.getRequestDispatcher("/error.jsp");
50         view.forward(request, response);
51         return;
52     }
53     // Business logic
54     DVDItem item = new DVDItem(title, year, genre);
55
56     // Store the item on the request-scope
57     request.setAttribute("dvdItem", item);
58
59     // Dispatch to Success view
60     RequestDispatcher view = request.getRequestDispatcher("/success.jsp");
61     view.forward(request, response);
62 }
```

Task 3: Remove the Old Servlet Code (optional)

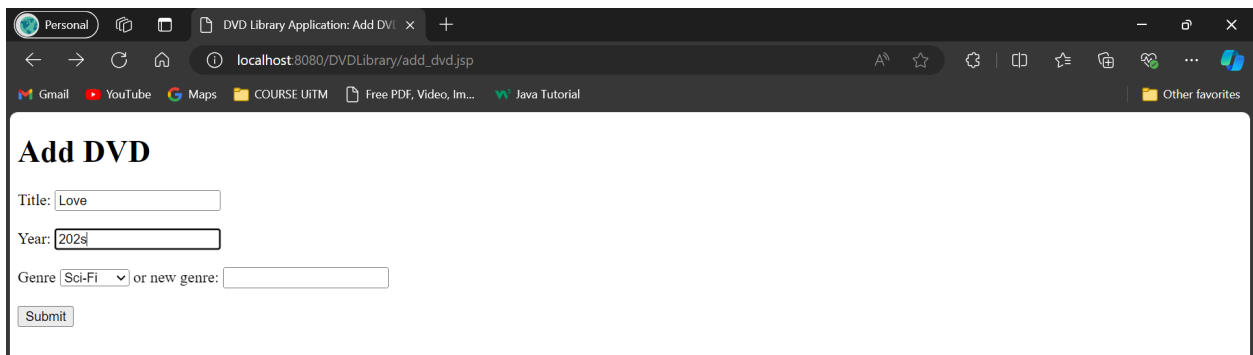
Task 4: Deploy and Test the Web Application



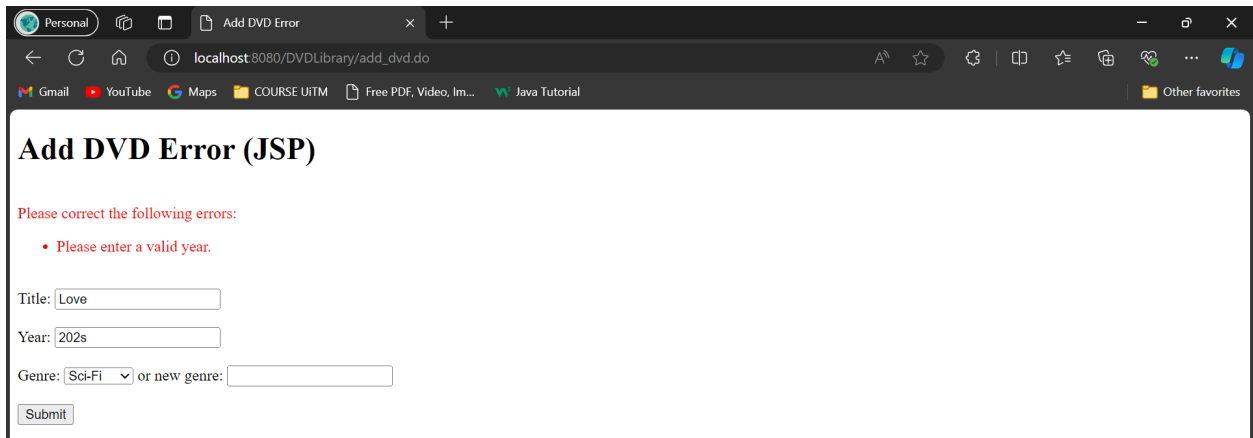
A screenshot of a web browser showing the 'Add DVD' form. The browser's address bar displays 'localhost:8080/DVDLibrary/add_dvd.jsp'. The form includes three input fields: 'Title' with the value 'Love', 'Year' with the value '2024', and 'Genre' with a dropdown menu set to 'Sci-Fi' and a text input for 'or new genre:'. A 'Submit' button is located at the bottom of the form.



A screenshot of a web browser showing the 'Add DVD Success (JSP)' page. The browser's address bar displays 'localhost:8080/DVDLibrary/add_dvd.do'. The page content states: 'You have add the following DVD: Title: Love Year Release: 2024 Genre of film: Sci-Fi'. Below this text is a link labeled 'Home'.



A screenshot of a web browser showing the 'Add DVD' form. The browser's address bar displays 'localhost:8080/DVDLibrary/add_dvd.jsp'. The form includes three input fields: 'Title' with the value 'Love', 'Year' with the value '202s', and 'Genre' with a dropdown menu set to 'Sci-Fi' and a text input for 'or new genre:'. A 'Submit' button is located at the bottom of the form.

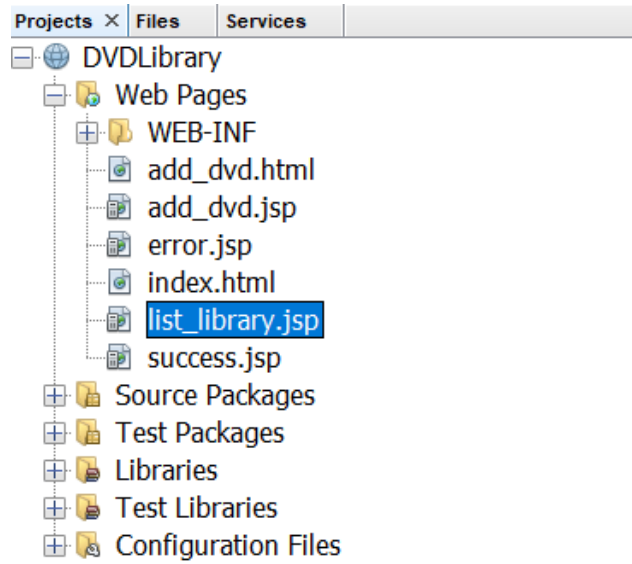


A screenshot of a web browser showing the 'Add DVD Error (JSP)' page. The browser's address bar displays 'localhost:8080/DVDLibrary/add_dvd.do'. The page content includes a red heading 'Add DVD Error (JSP)', a red message 'Please correct the following errors:', and a red bullet point 'Please enter a valid year:'. Below this is the 'Add DVD' form with 'Title' as 'Love', 'Year' as '202s', and 'Genre' as 'Sci-Fi'. A 'Submit' button is at the bottom.

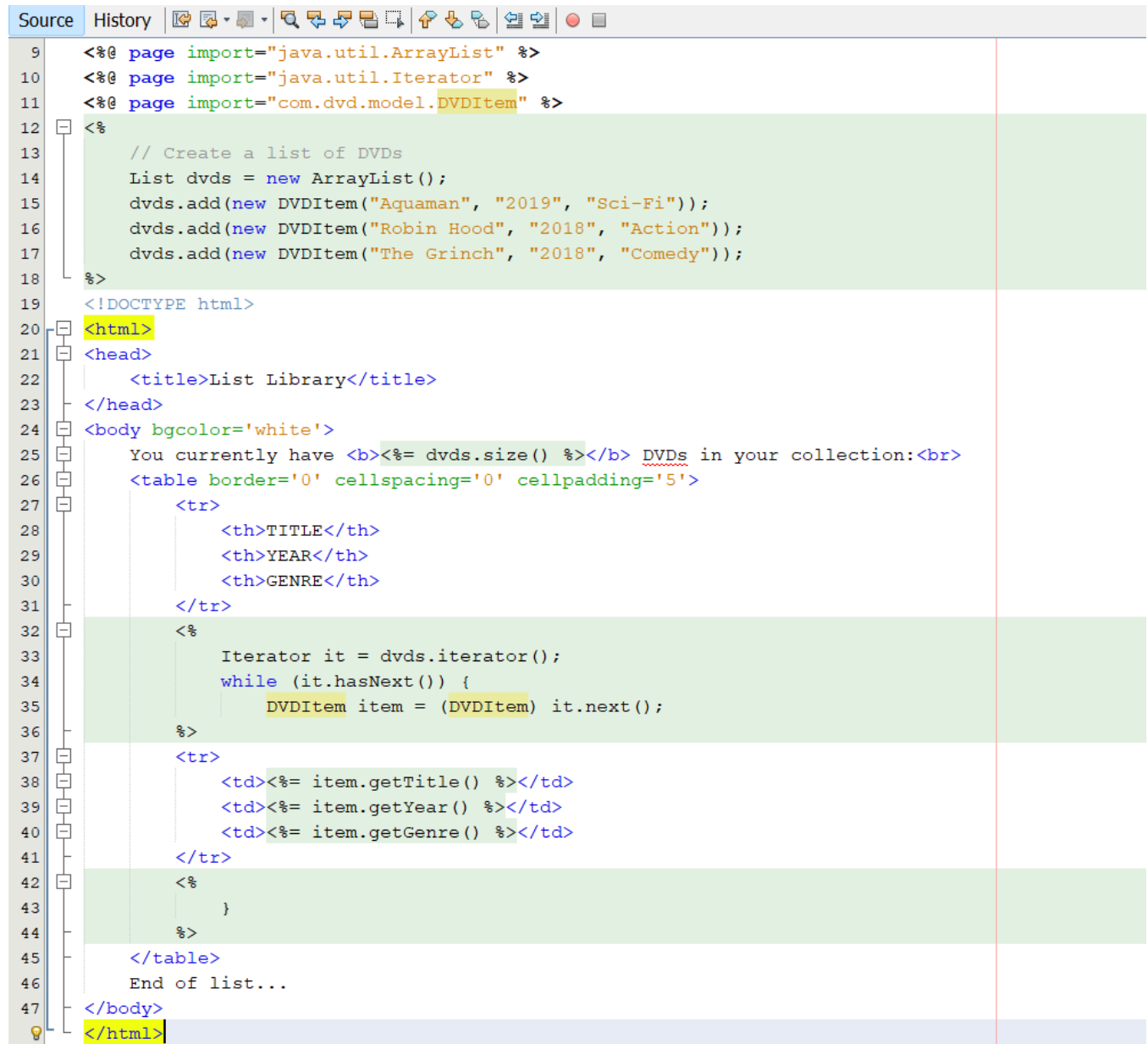
Exercise 3: Converting the List Library View Component to a JSP Page

Task 1: Convert the List Library Servlet

1. Create a JSP page file with the following characteristics:

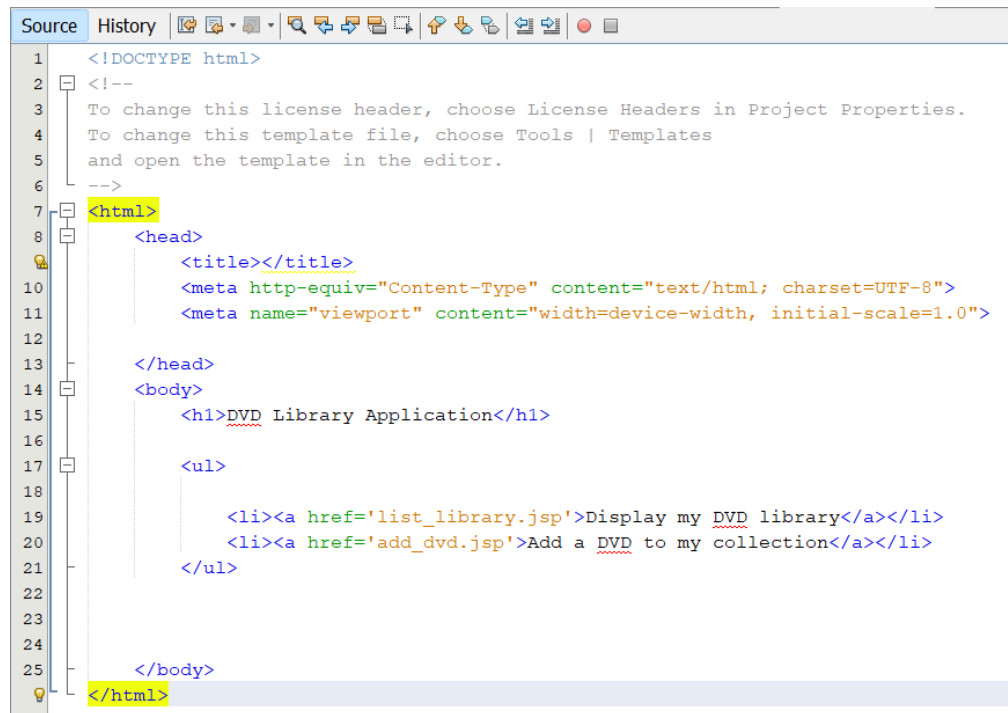


2. Using the ListLibraryServlet servlet class as a guide, convert the servlet code into JSP code.



```
9  <%@ page import="java.util.ArrayList" %>
10 <%@ page import="java.util.Iterator" %>
11 <%@ page import="com.dvd.model.DVDItem" %>
12 <%
13     // Create a list of DVDs
14     List dvds = new ArrayList();
15     dvds.add(new DVDItem("Aquaman", "2019", "Sci-Fi"));
16     dvds.add(new DVDItem("Robin Hood", "2018", "Action"));
17     dvds.add(new DVDItem("The Grinch", "2018", "Comedy"));
18 %>
19 <!DOCTYPE html>
20 <html>
21 <head>
22     <title>List Library</title>
23 </head>
24 <body bgcolor='white'>
25     You currently have <b><%= dvds.size() %></b> DVDs in your collection:<br>
26     <table border='0' cellspacing='0' cellpadding='5'>
27         <tr>
28             <th>TITLE</th>
29             <th>YEAR</th>
30             <th>GENRE</th>
31         </tr>
32         <%
33             Iterator it = dvds.iterator();
34             while (it.hasNext()) {
35                 DVDItem item = (DVDItem) it.next();
36             %>
37         <tr>
38             <td><%= item.getTitle() %></td>
39             <td><%= item.getYear() %></td>
40             <td><%= item.getGenre() %></td>
41         </tr>
42         <%
43             }
44         %>
45     </table>
46     End of list...
47 </body>
48 </html>
```

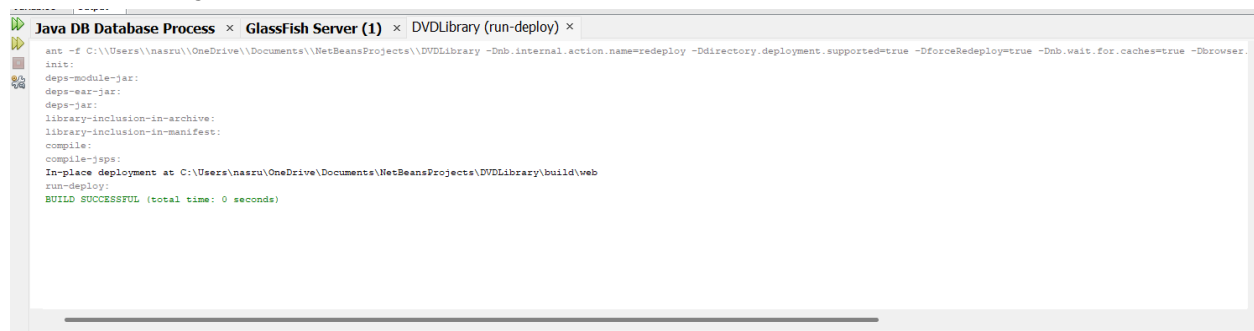
3. Modify the index.html home page to change the link to the new List Library JSP page.



```
1 <!DOCTYPE html>
2 <!--
3 To change this license header, choose License Headers in Project Properties.
4 To change this template file, choose Tools | Templates
5 and open the template in the editor.
6 -->
7 <html>
8 <head>
9 <title></title>
10 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
11 <meta name="viewport" content="width=device-width, initial-scale=1.0">
12
13 </head>
14 <body>
15 <h1>DVD Library Application</h1>
16
17 <ul>
18
19 <li><a href='list_library.jsp'>Display my DVD library</a></li>
20 <li><a href='add_dvd.jsp'>Add a DVD to my collection</a></li>
21 </ul>
22
23
24
25 </body>
26 </html>
```

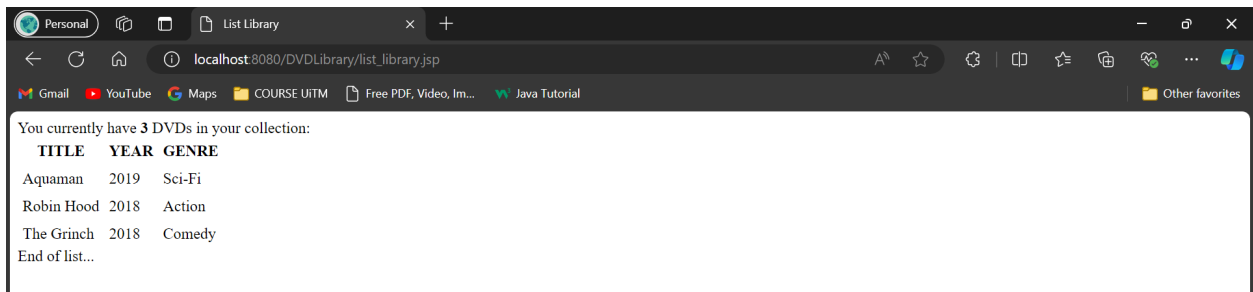
Task 2: Remove the Old Servlet Code (optional)

Task 3: Deploy the Web Application

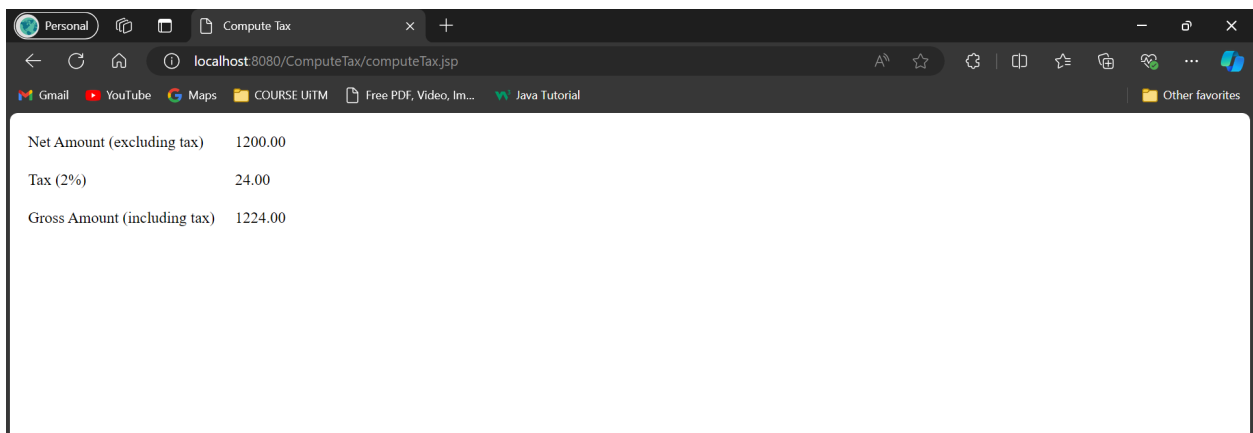
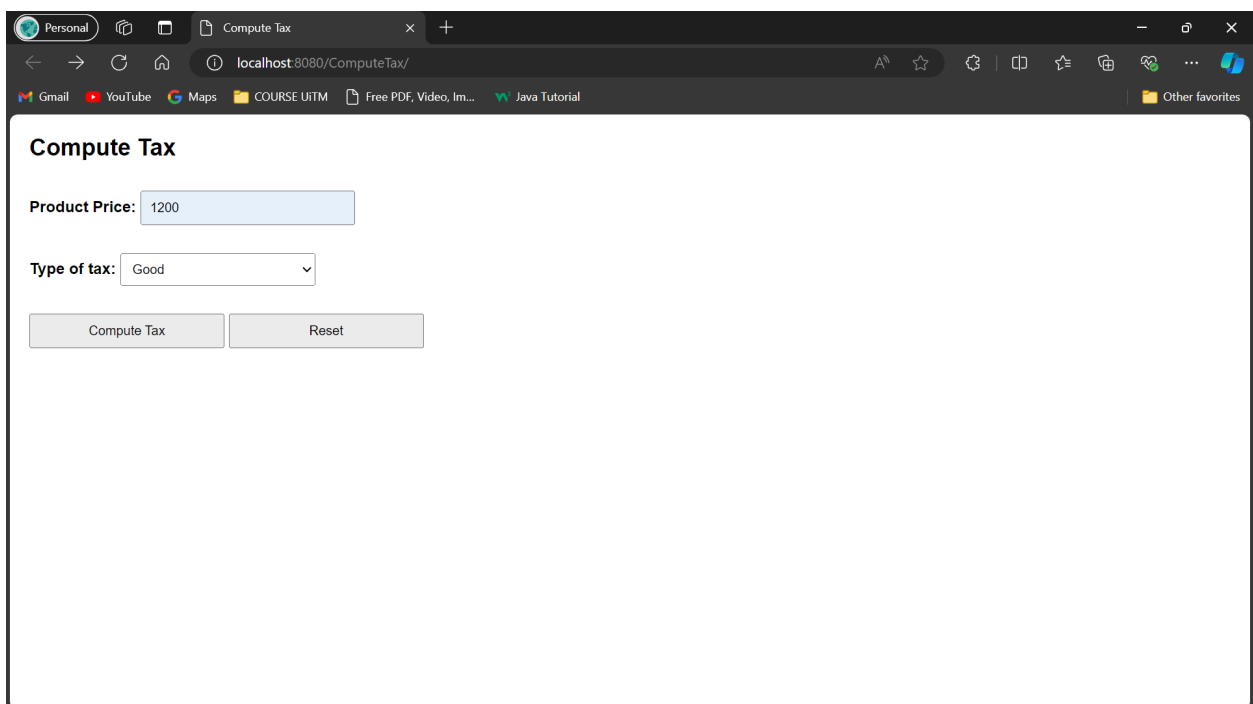


```
Java DB Database Process x GlassFish Server (1) x DVDLibrary (run-deploy) x
ant -f C:\Users\nasru\OneDrive\Documents\NetBeansProjects\DVDLibrary -Dnb.internal.action.name=redeploy -Ddirectory.deployment.supported=true -Dforce.redeploy=true -Dnb.wait.for.caches=true -Dbrowser.
init:
deps-module-jar:
deps-war-jar:
deps-jar:
library-inclusion-in-archive:
library-inclusion-in-manifest:
compile:
compile-jsp:
In-place deployment at C:\Users\nasru\OneDrive\Documents\NetBeansProjects\DVDLibrary\build\web
run-deploy:
BUILD SUCCESSFUL (total time: 0 seconds)
```

Task 4: Test the Web Application



Post Lab Exercise



LAB 6: Database Programming in Servlets

Lab Activities

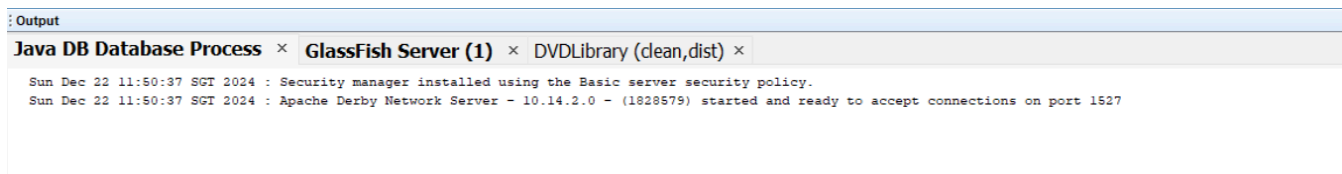
1. Create a database and a table in Java Derby

Task 1: Starting the Server and Creating a Database

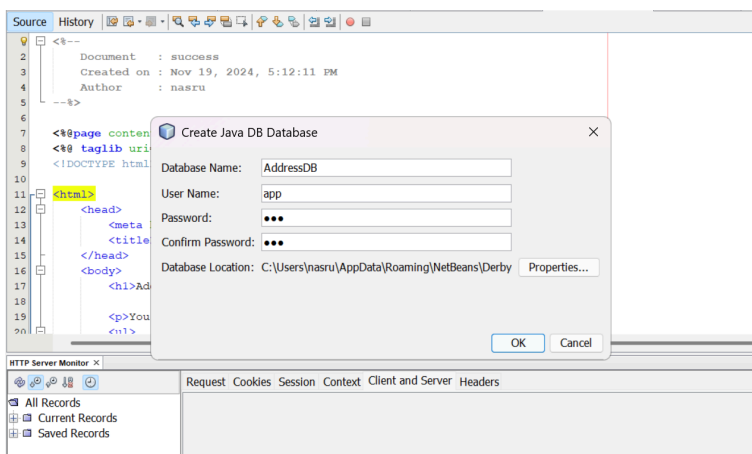
The Java DB Database menu options are displayed when you right-click the Java DB node in the Services window. This contextual menu items allow you to start and stop the database server, create a new database instance, as well as register database servers in the IDE (as demonstrated in the previous step).

To start the database server:

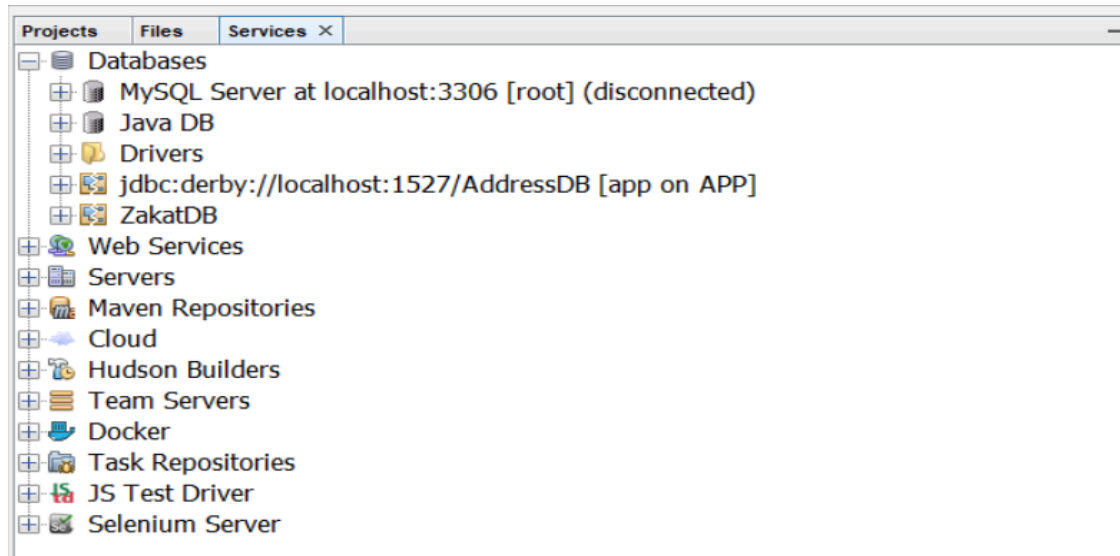
- In the **Services** window, right-click the **Java DB** node and choose **Start Server**. Note the following output in the Output window, indicating that the server has started:



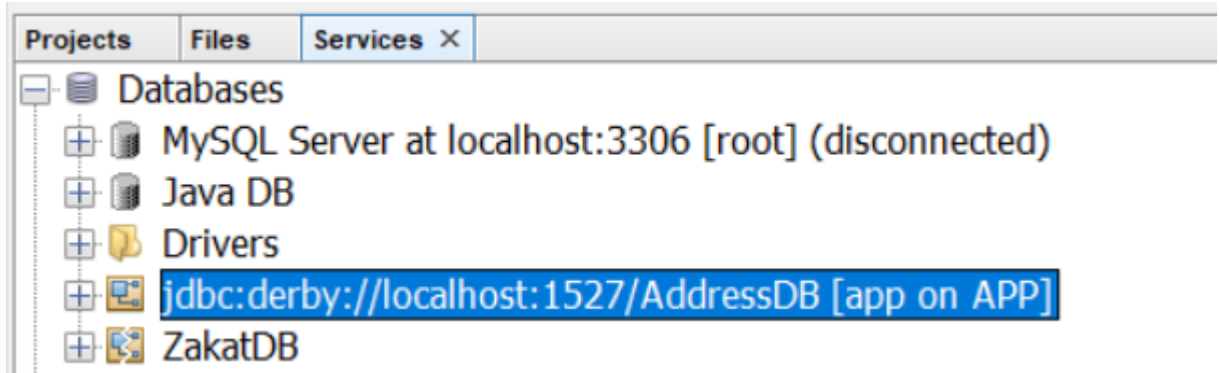
- Right-click the **Java DB** node and choose **Create Database** to open the Create Java DB Database dialog.
- Type **AddressDB** for the Database Name.
- Type **app** for the User Name and Password. Click OK.



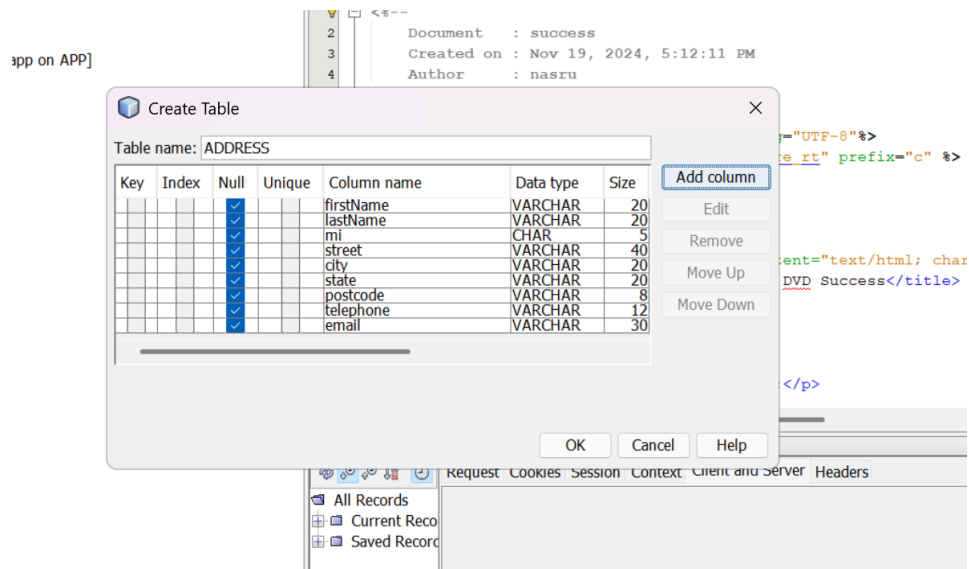
After you create the database, if you expand the Databases node in the Services window you can see that the IDE created a database connection and that the database was added to the list under the Java DB node.




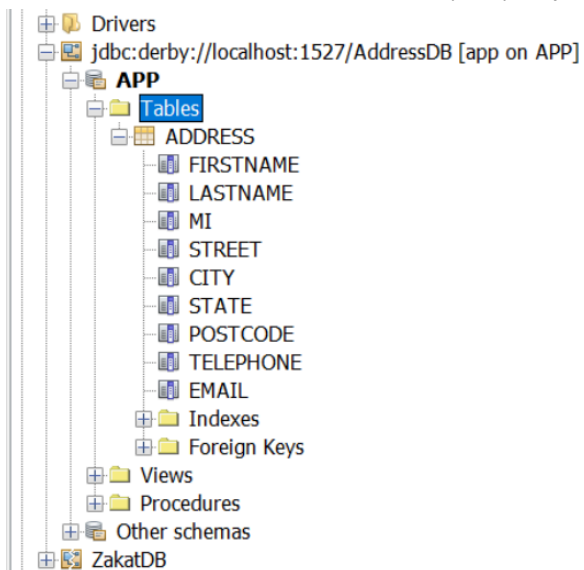
Task 2: Connecting to the Database



Task 3) Creating Tables

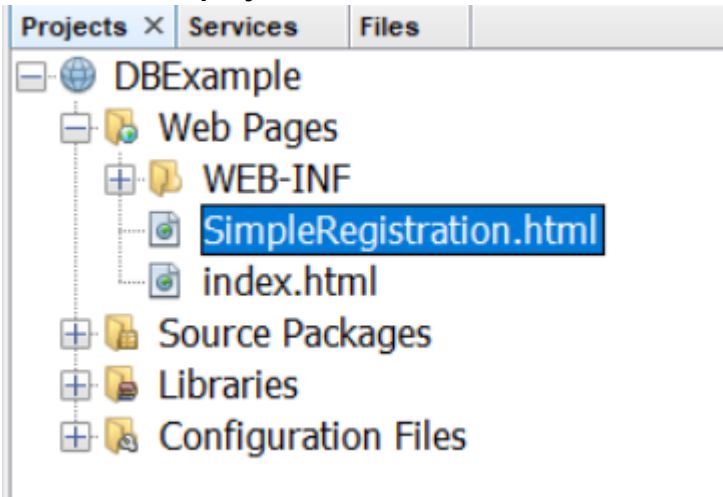


- f. When you are sure that your Create Table dialog contains the same specifications as those shown above, click OK. The IDE generates the ADDRESS table in the database, and you can see a new ADDRESS table node () display under the Tables node.



2. Creating web components.

Task 1: Create a web project



Task 2: Create an HTML file

- Create an HTML file name **SimpleRegistration.html** for collecting the data and sending it to the database using the post method.

```

9      <title>Simple Registration without confirmation</title>
10     <meta charset="UTF-8">
11     <meta name="viewport" content="width=device-width, initial-scale=1.0">
12 </head>
13 <body>
14     Please register to your instructor's student address book
15     <form method = "post" action = " /DBExample/SimpleRegistration">
16         <p>Last Name <font color="#FF0000">*</font>
17             <input type="text" name="lastName">&nbsp;
18             First Name <font color="#FF0000">*</font>
19             <input type="text" name="firstName">&nbsp;
20             MI <input type="text" name="mi" size="3">
21         </p>
22         <p>Telephone
23             <input type="text" name="telephone" size="20">&nbsp;
24             Email
25             <input type="text" name="email" size="30">&nbsp;
26         </p>
27         <p>Street <input type="text" name="street" size="40"></p>
28         <p>
29             City <input type="text" name="city" size="20">&nbsp;
30             Postcode <input type="text" name="postcode" size="6">
31         </p>
32         <p>
33             State
34             <select size="1" name="state">
35                 <option value="Selangor">Selangor</option>
36                 <option value="Perak">Perak</option>
37                 <option value="Kedah">Kedah</option>
38                 <option value="Melaka">Melaka</option>
39                 <option value="Johor">Johor</option>
40                 <option value="Negeri Sembilan">Negeri Sembilan</option>
41             </select>&nbsp;
42         </p><br>

```

Task 3: Create a Servlet

- a. Create a Java Servlet with the following characteristic: Class name: **SimpleRegistration**

```

index.html x SimpleRegistration.html x SimpleRegistration.java x
Source History
7  import java.io.IOException;
8  import java.io.PrintWriter;
9  import javax.servlet.ServletException;
10 import javax.servlet.http.HttpServlet;
11 import javax.servlet.http.HttpServletRequest;
12 import javax.servlet.http.HttpServletResponse;
13 import java.sql.*;
14
15
16 /**
17  *
18  * @author nasru
19  */
20 public class SimpleRegistration extends HttpServlet {
21     private PreparedStatement pstmt;
22     private Connection conn;
23
24     public void init() throws ServletException{
25         initializeJdbc();
26     }

```

```

index.html x SimpleRegistration.html x SimpleRegistration.java x
Source History
27 @Override
28 protected void doPost(HttpServletRequest request, HttpServletResponse response)
29     throws ServletException, IOException {
30     response.setContentType ("text/html");
31     PrintWriter out = response.getWriter();
32
33     String lastName = request.getParameter("lastName");
34     String firstName = request.getParameter("firstName");
35     String mi = request.getParameter("mi");
36     String phone = request.getParameter("phone");
37     String email = request.getParameter("email");
38     String address = request.getParameter("address");
39     String city = request.getParameter("city");
40     String postcode = request.getParameter("postcode");
41     String state = request.getParameter("state");
42
43     try {
44         if (lastName.length() == 0 || firstName.length() == 0){
45             out.println("Last Name and First name are required");
46             return;
47         }
48         storeStudent(lastName,firstName,mi,phone,email,address,city,state,postcode);
49         out.println(firstName+" "+lastName+" is now registered in the database");
50     } catch (Exception ex) {
51         out.println("Error: "+ex.getMessage());
52     } finally {
53         out.close();
54     }
55 }
56

```

```
Index.html x SimpleRegistration.html x SimpleRegistration.java x
Source History
57 private void initializeJdbc() {
58     try {
59         String driver = "org.apache.derby.jdbc.ClientDriver";
60         String connectionString = "jdbc:derby://localhost:1527/AddressDB";
61         String usr="app", pass="app";
62
63         Class.forName(driver);
64
65         conn = DriverManager.getConnection(connectionString,usr,pass);
66
67     } catch (Exception ex) {
68         ex.printStackTrace();
69     }
70 }
71
72 private void storeStudent(String lastName,String firstName,String mi,String phone,String email,String address,String city,String state,String post
73     String sql = "insert into Address "
74         + "(lastName,firstName,mi,phone,email,address,city,"
75         + "state,postcode) value (?, ?, ?, ?, ?, ?, ?, ?)";
76     pstmt = conn.prepareStatement(sql);
77     pstmt.setString(1, lastName);
78     pstmt.setString(2, firstName);
79     pstmt.setString(3, mi);
80     pstmt.setString(4, phone);
81     pstmt.setString(5, email);
82     pstmt.setString(6, address);
83     pstmt.setString(7, city);
84     pstmt.setString(8, state);
85     pstmt.setString(9, postcode);
86
87     pstmt.executeUpdate();
88 }
```

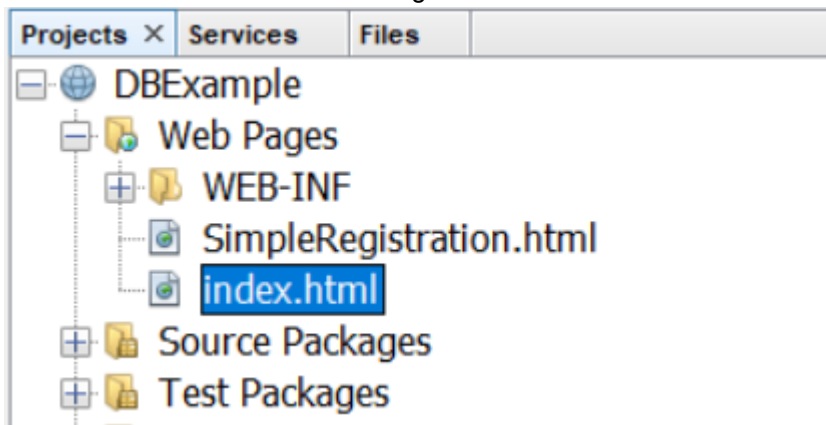
3. Configuring the Web application

Task 1 : Verify the view servlet configuration

```
index.html x SimpleRegistration.html x SimpleRegistration.java x web.xml x
Source General Servlets Filters Pages References Security History
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xmlns:xsi="http
3     <servlet>
4         <servlet-name>SimpleRegistration</servlet-name>
5         <servlet-class>SimpleRegistration</servlet-class>
6     </servlet>
7     <servlet-mapping>
8         <servlet-name>SimpleRegistration</servlet-name>
9         <url-pattern>/SimpleRegistration</url-pattern>
10    </servlet-mapping>
11    <session-config>
12        <session-timeout>
13            30
14        </session-timeout>
15    </session-config>
16 </web-app>
17
```

Task 2: Create the homepage

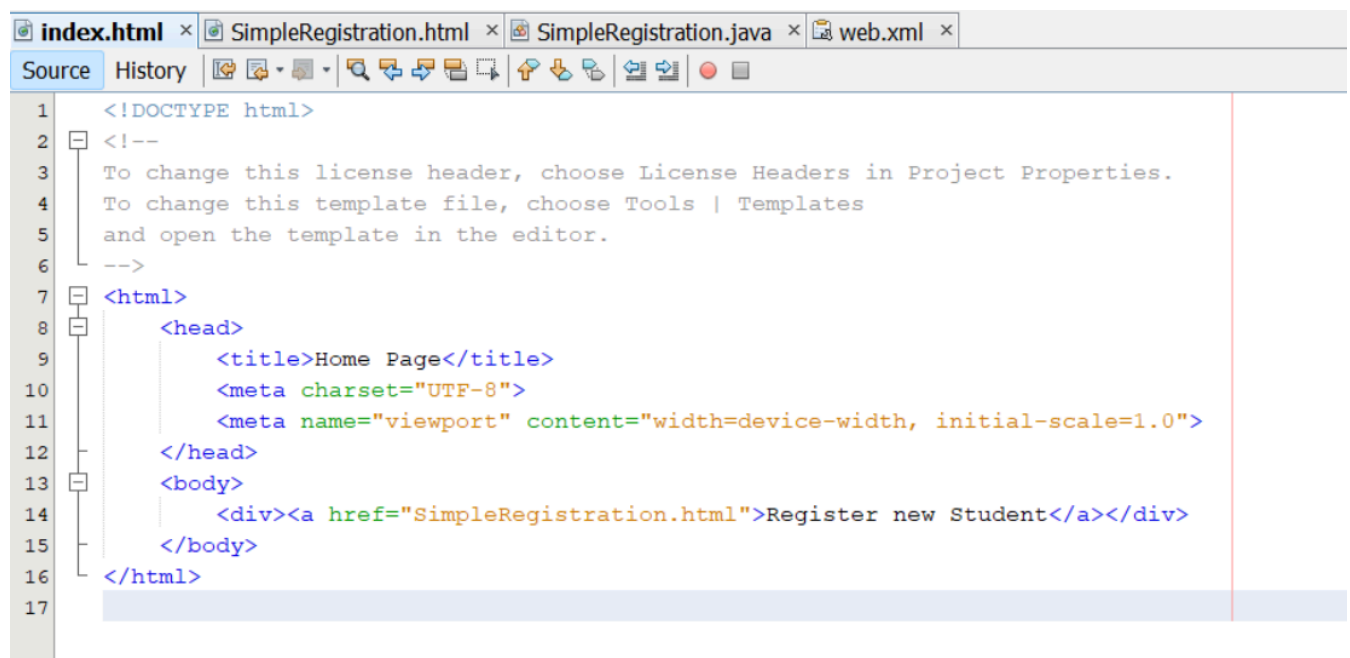
a. Create an HTML file with the following characteristic:



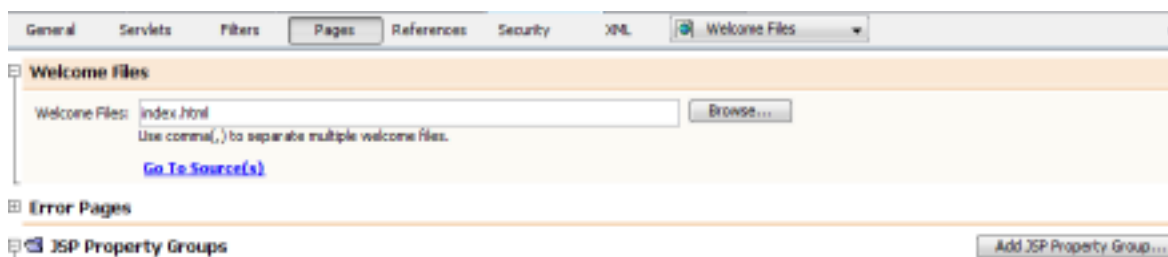
b. Edit the index.html file so that it displays the following text.

Register new student

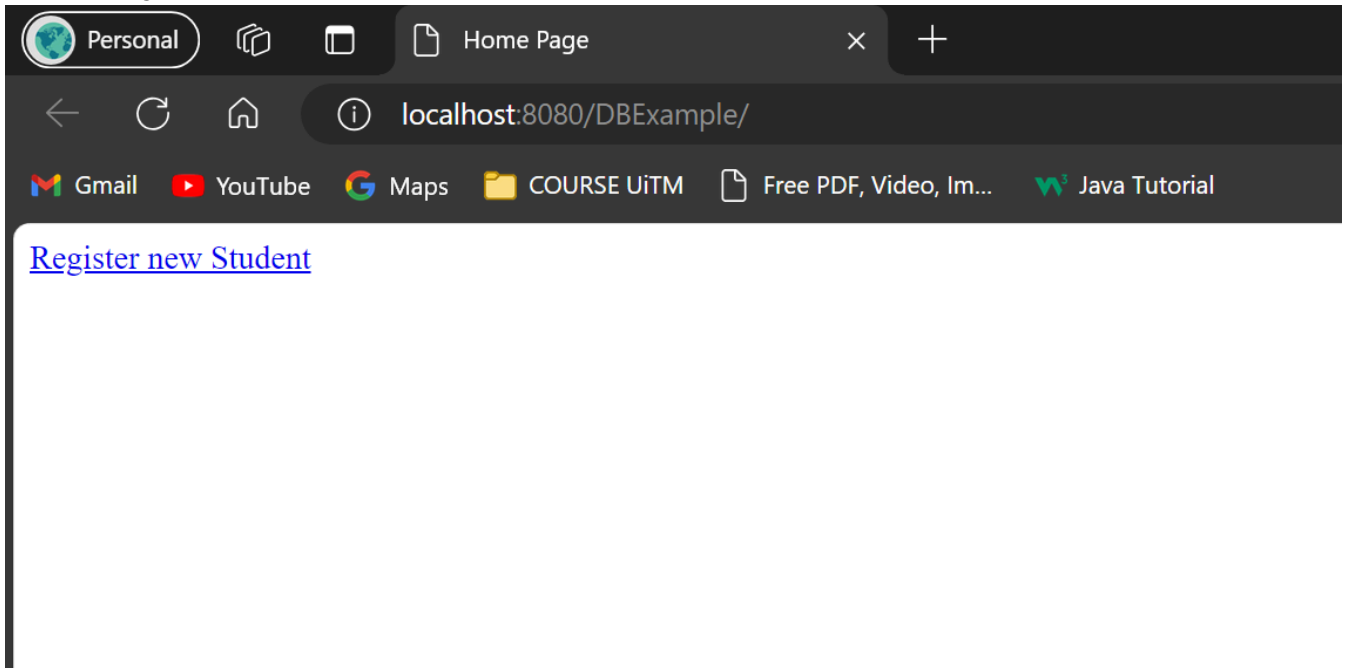
Edit the index.html file to add a link to the SimpleRegistration.html page.



c. Open the web.xml deployment descriptor and set index.html as the welcome file.



Task 3: Deploy the web application



A screenshot of a web browser window displaying a registration form. The address bar shows 'localhost:8080/DBExample/SimpleRegistration.html'. The form is titled 'Please register to your instructor's student address book'. It contains several input fields: 'Last Name *', 'First Name *', 'MI', 'Telephone', 'Email', 'Street', 'City', 'Postcode', and a 'State' dropdown menu currently set to 'Selangor'. At the bottom of the form are 'Submit' and 'Reset' buttons. Below the buttons, a red asterisk is followed by the text '* Required Fields'.

Please register to your instructor's student address book

Last Name * First Name * MI

Telephone Email

Street

City Postcode

State

* Required Fields

Personal Simple Registration without confi x

localhost:8080/DBExample/SimpleRegistration.html

Gmail YouTube Maps COURSE UiTM Free PDF, Video, Im... Java Tutorial

Please register to your instructor's student address book

Last Name * SHAHDAN First Name * NASRULLAH MI A

Telephone 01140176845 Email nasrullahshahdan@gmail.com

Street KAMPUNG SEBERANG JELAI

City KUALA LIPIS Postcode 27100

State Kedah

Submit Reset

* Required Fields

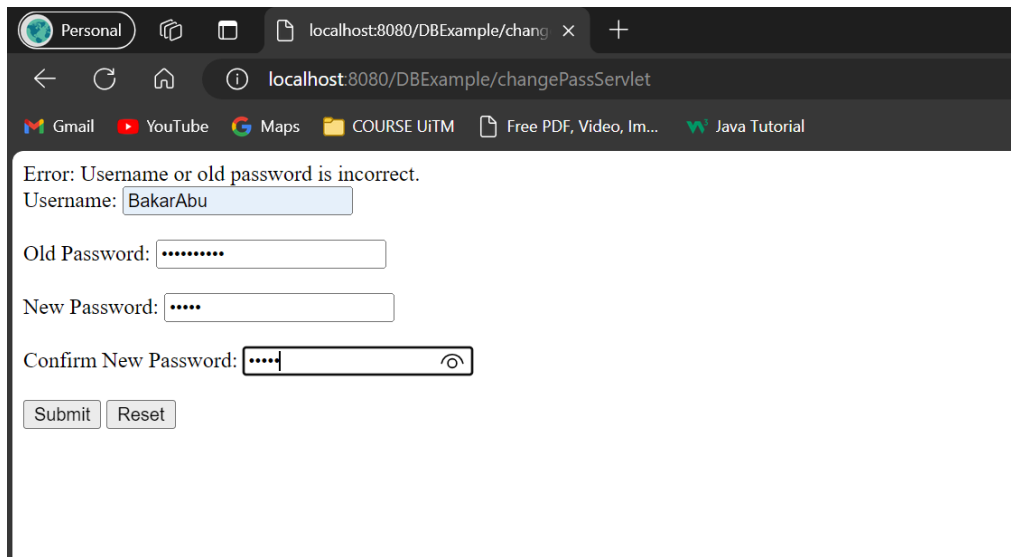
Personal localhost:8080/DBExample/Simple x

localhost:8080/DBExample/SimpleRegistration

Gmail YouTube Maps COURSE UiTM Free PDF, Video, Im... Java Tutorial

NASRULLAH SHAHDAN is now registered in the database

Postlab Exercise



A screenshot of a web browser window. The address bar shows 'localhost:8080/DBExample/changePassServlet'. The page displays an error message: 'Error: Username or old password is incorrect.' Below the error, there is a form with the following fields: 'Username:' with the value 'BakarAbu', 'Old Password:' with masked characters '.....', 'New Password:' with masked characters '....', and 'Confirm New Password:' with masked characters '....'. At the bottom of the form are two buttons: 'Submit' and 'Reset'.

