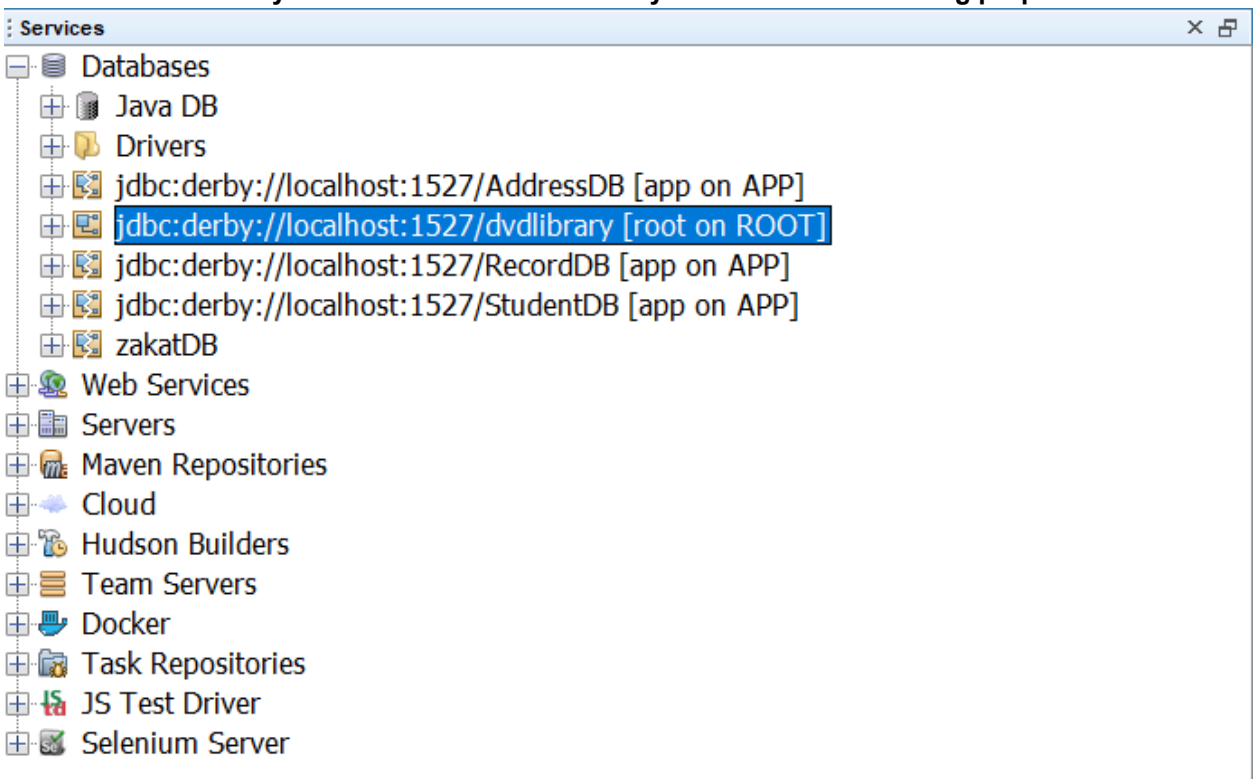
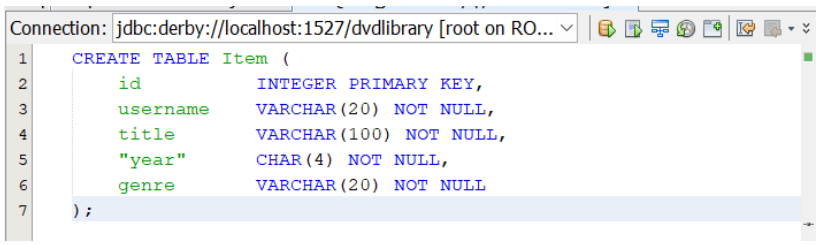
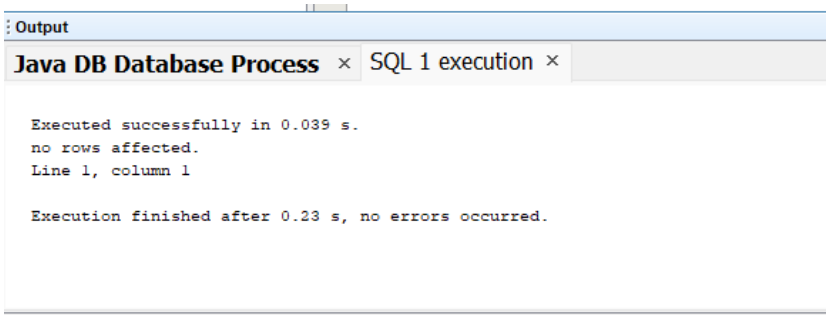
Post LAB 9: Integrating JSP with a database

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 ShanwsShark 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');
```

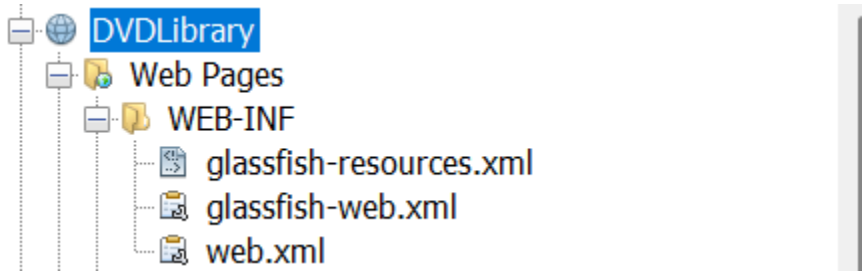
Output

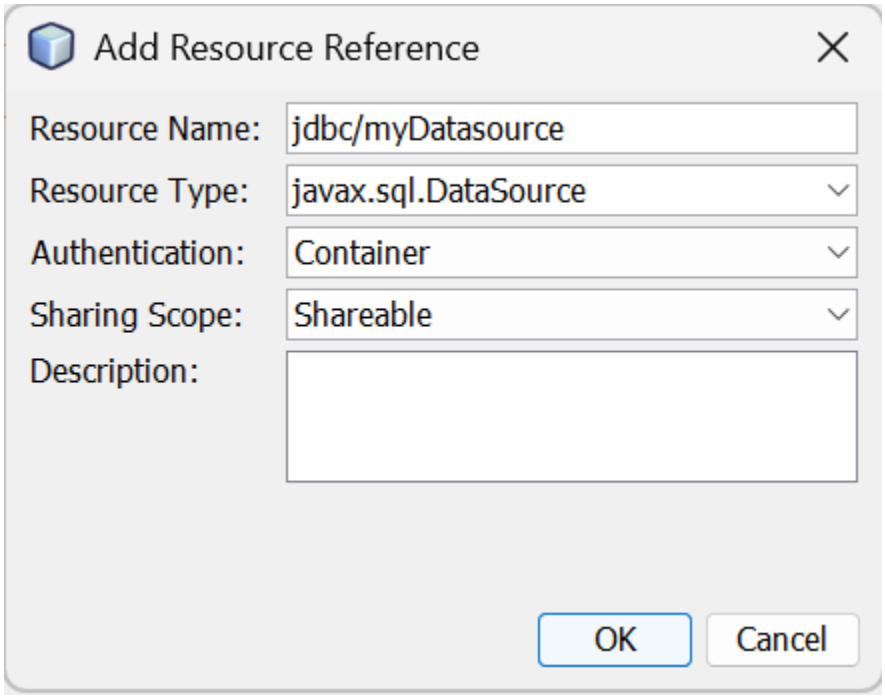
Java DB Database Process | SQL 1 execution

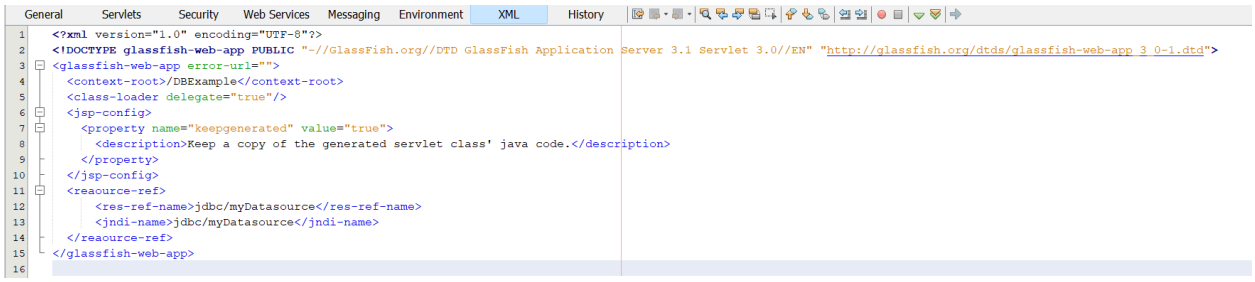
```
1 row affected.
Line 7, column 1


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:

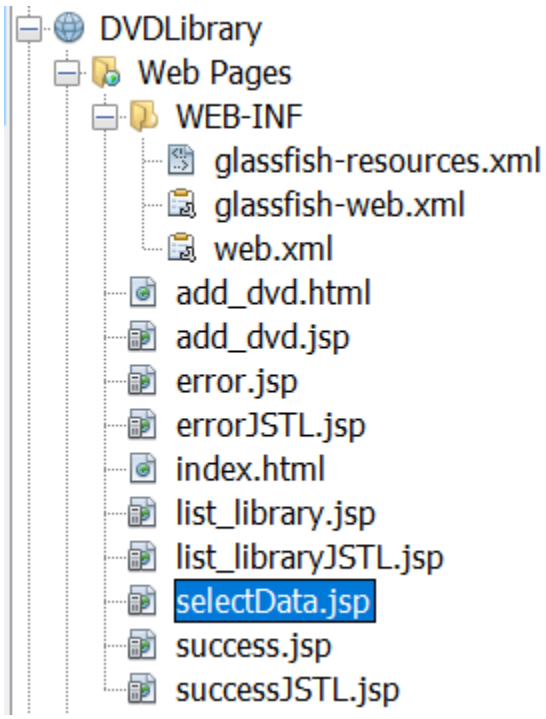
1. Configure the Data source in the deployment descriptor
2. Configure the data source in the application server

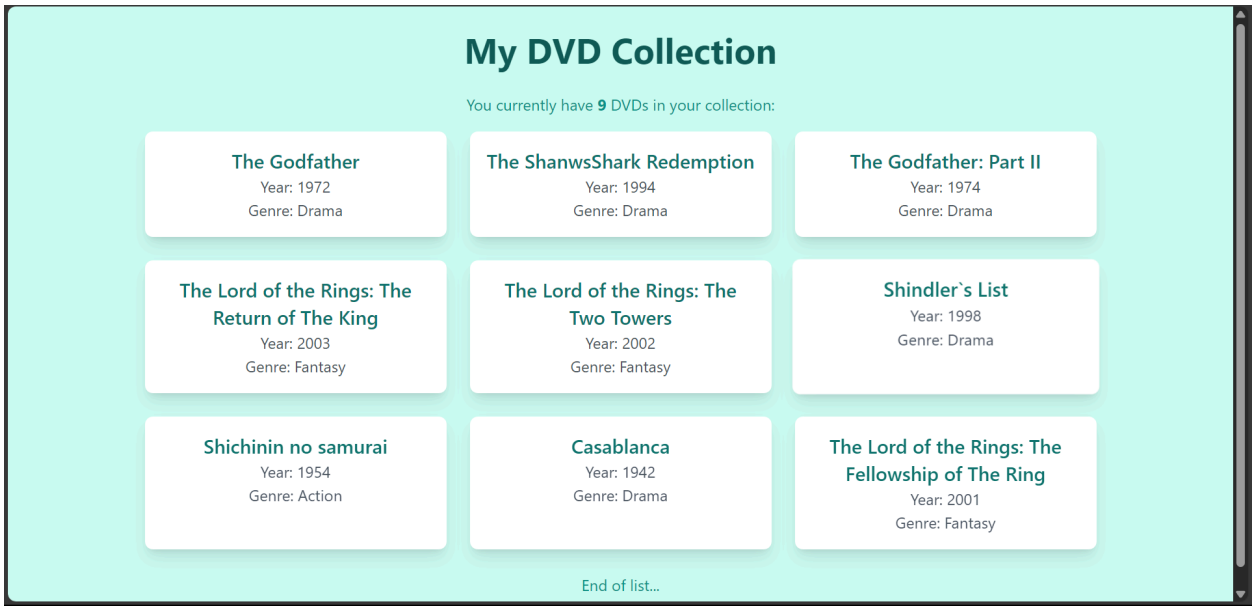






3. Create a new JSP file as selectData.jsp. Write JSP/ JSTL code to display the data from database as follow:



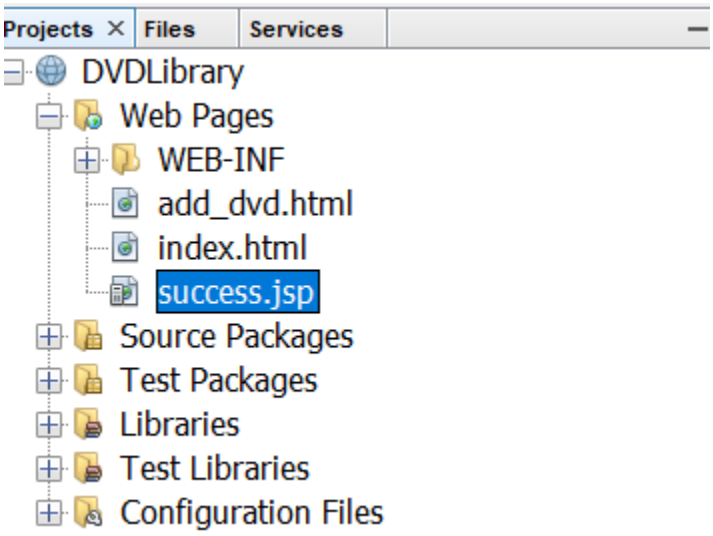


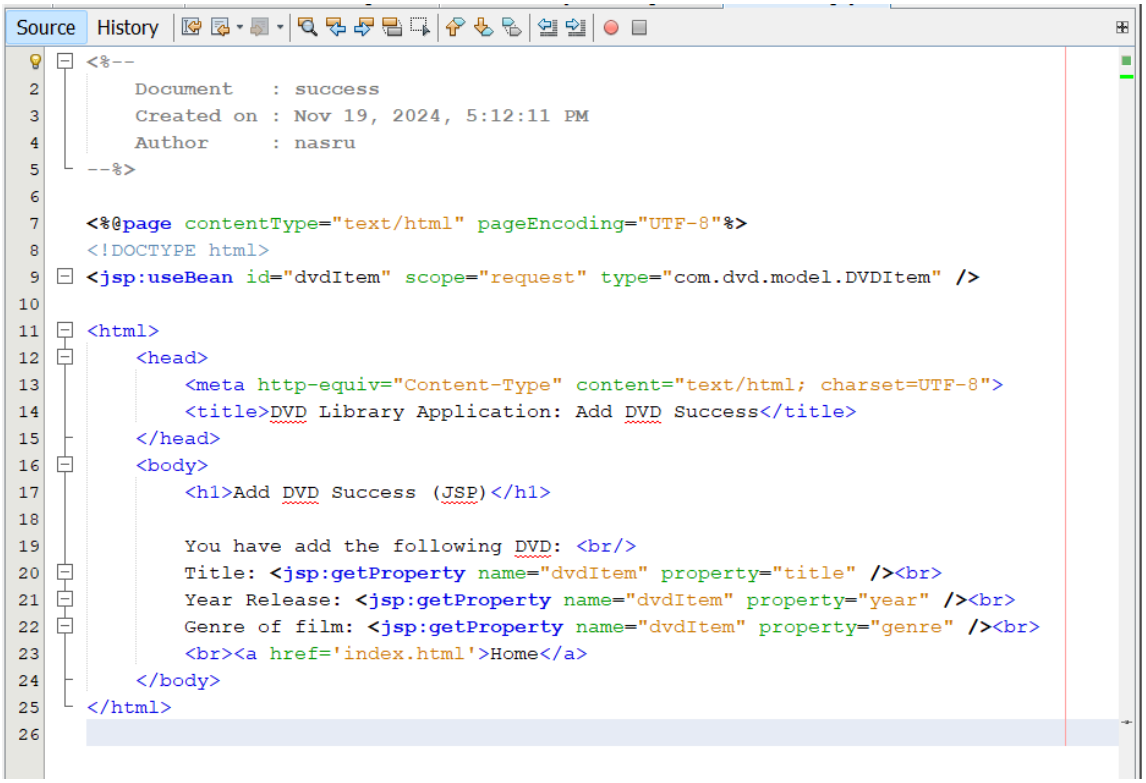
LAB 7: JSP - CSC584

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:



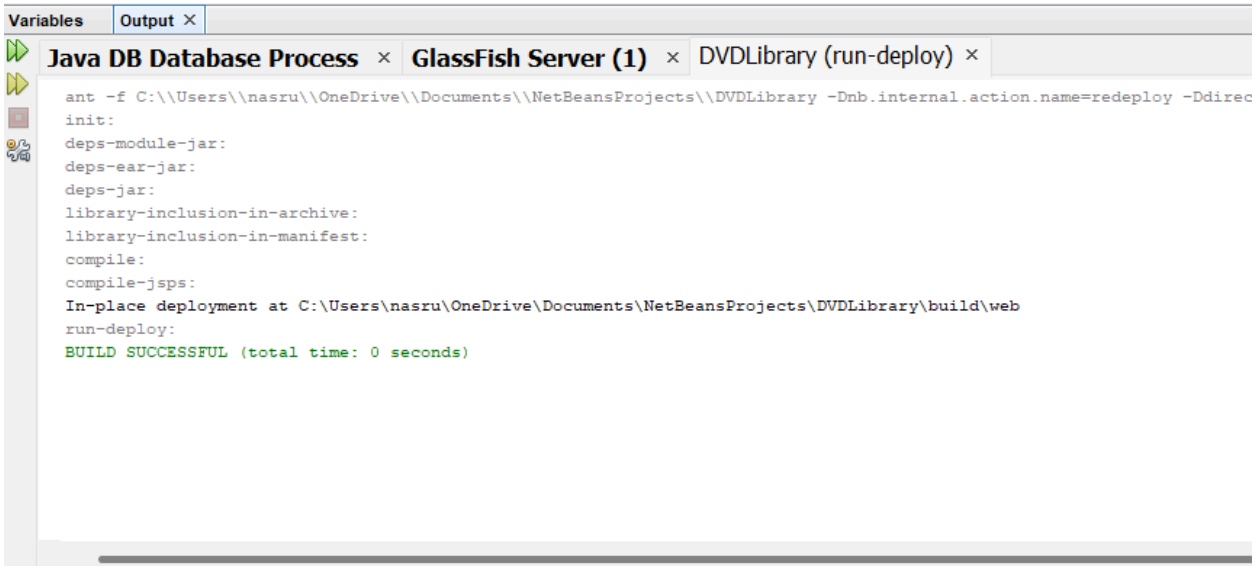


Task 2: Remove the Old Servlet Code (optional)

Source | General | Servlets | Filters | Pages | References | Security | History

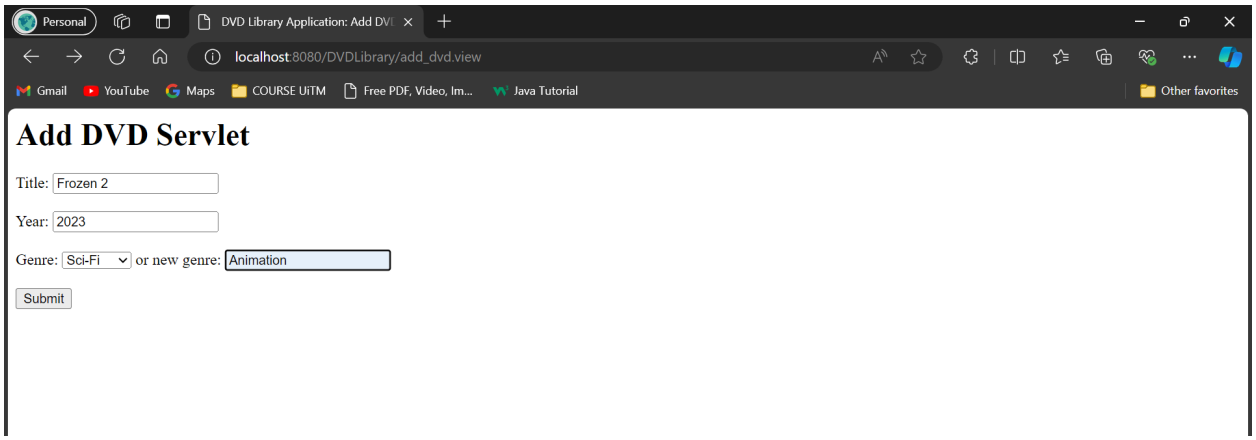
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xmlns:xsi="http
    <servlet>
        <servlet-name>ListLibraryServlet</servlet-name>
        <servlet-class>com.dvd.view.ListLibraryServlet</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>AddDVDServlet</servlet-name>
        <servlet-class>com.dvd.controller.AddDVDServlet</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>AddDVDFormServlet</servlet-name>
        <servlet-class>com.dvd.view.AddDVDFormServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>ListLibraryServlet</servlet-name>
        <url-pattern>/list_library.view</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>AddDVDServlet</servlet-name>
        <url-pattern>/add_dvd.do</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>AddDVDFormServlet</servlet-name>
        <url-pattern>/add_dvd.view</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
    </welcome-file-list>
</web-app>
```

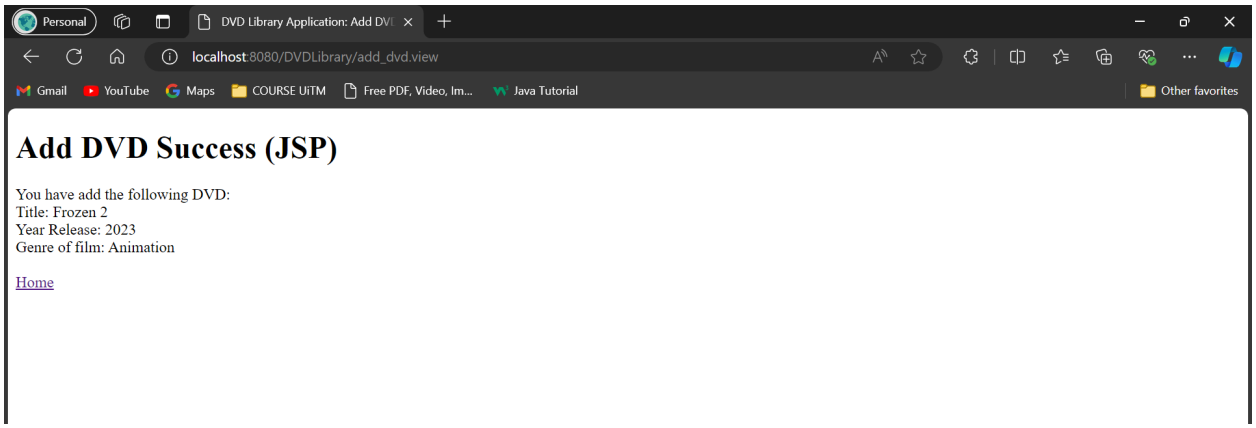
Task 3: Deploy the Web Application



Task 4: Test the Web Application

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

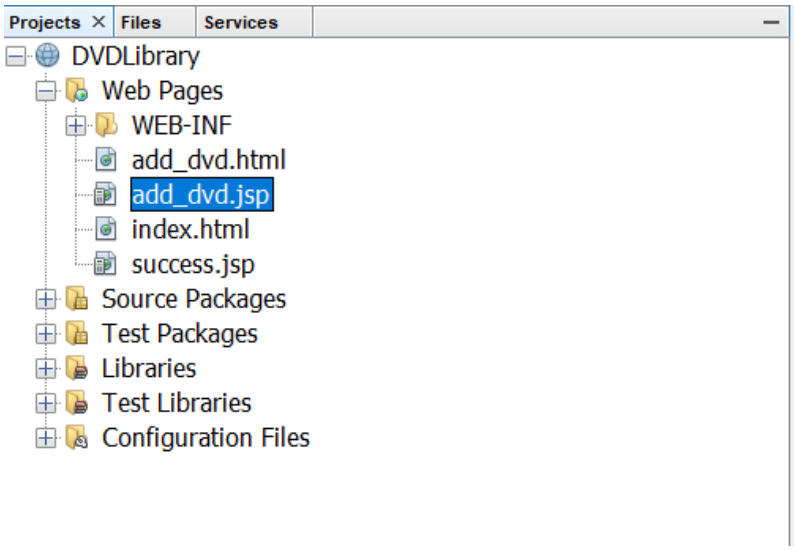




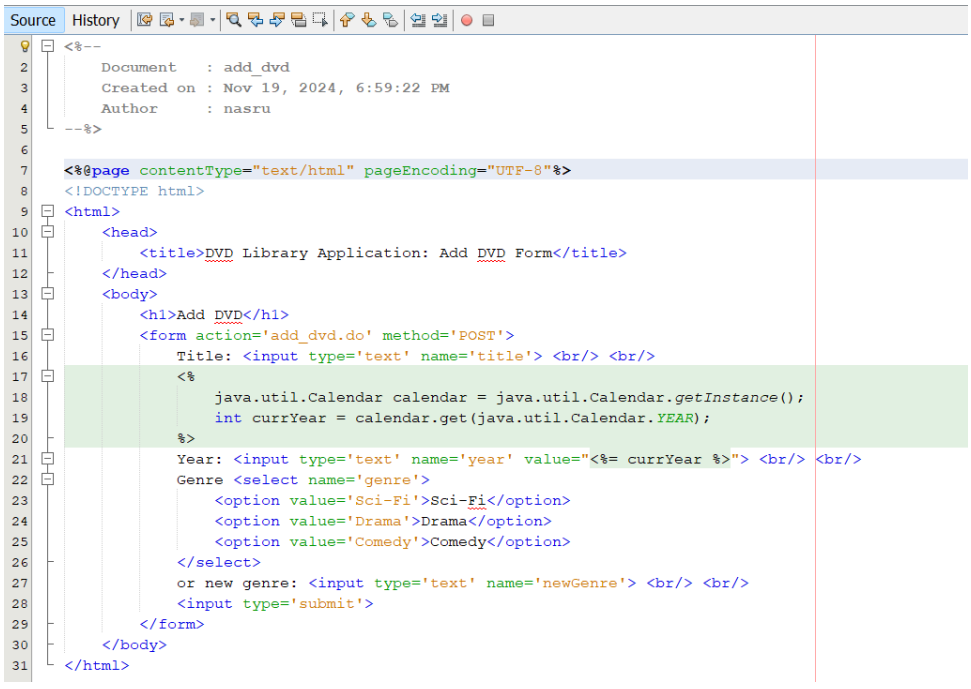
Exercise 2: Converting the Add DVD Form Component to a JSP Page

Task 1: Convert the AddDVDFormServlet Servlet

1.



2. + 3

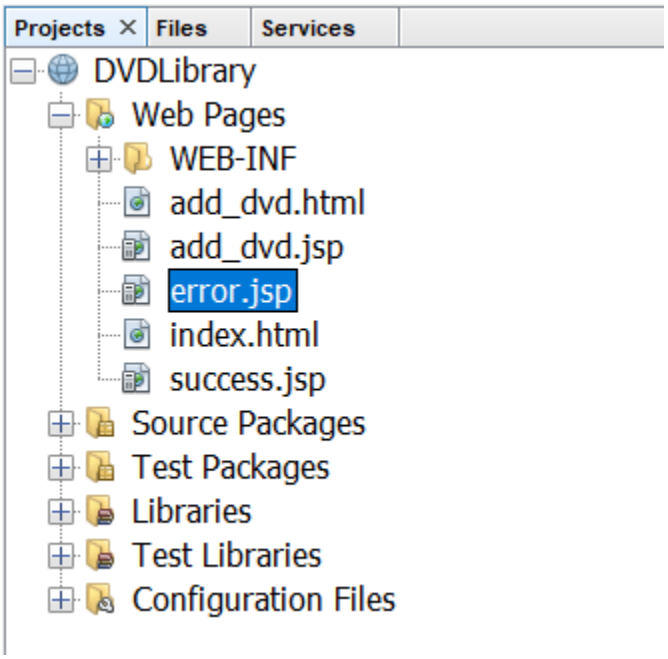


4.

```
Source  History
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
25     </body>
26 </html>
```

Task 2: Create the JSP error page

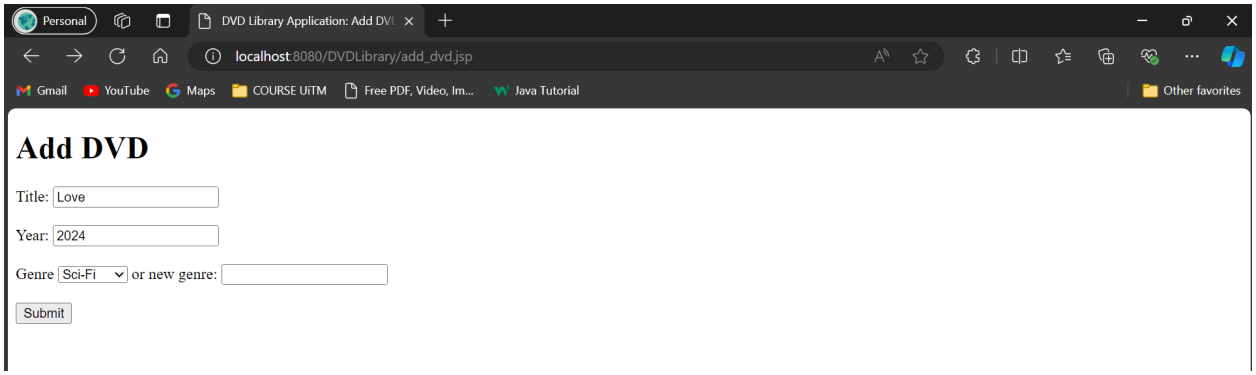
1. Create JSP file with the following characteristics:

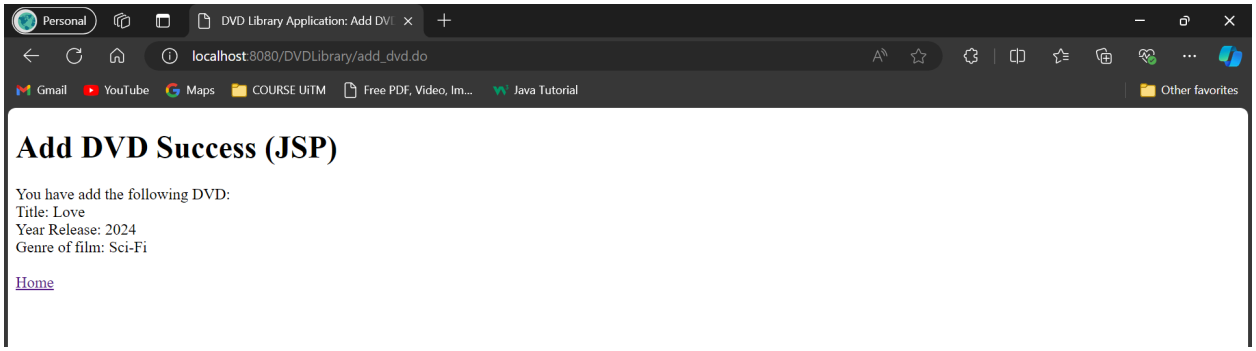


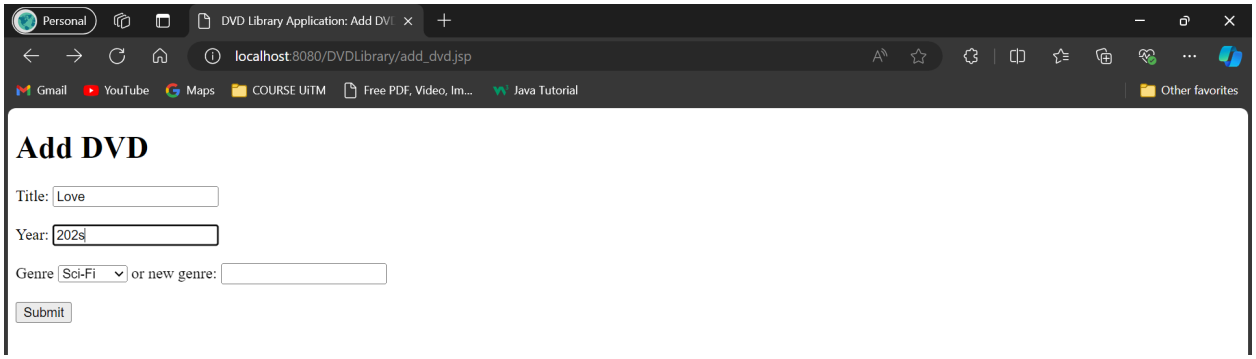
2. + 3

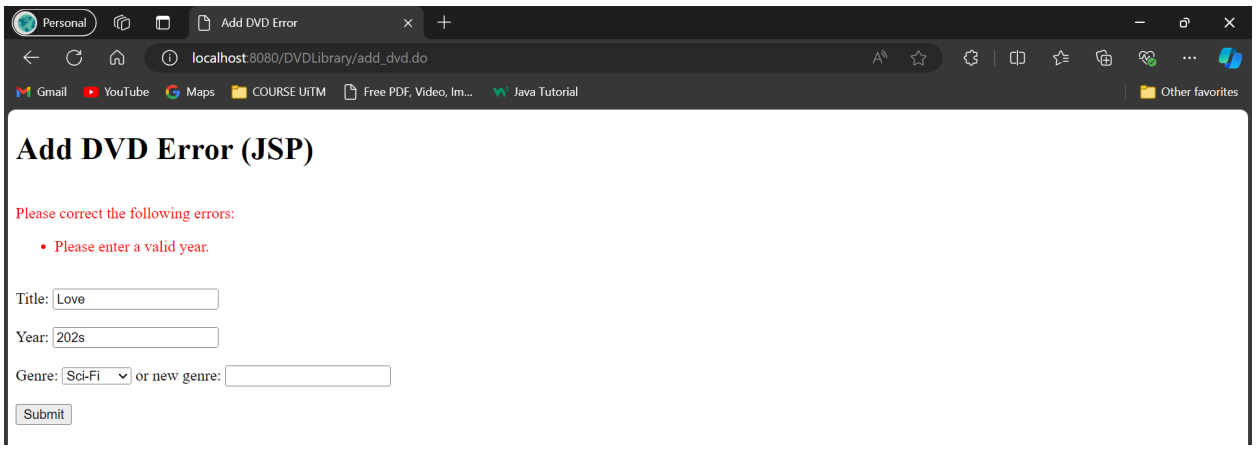
```
46              }
47              if(!errorMsgs.isEmpty()){
48                  request.setAttribute("errorMsgs", errorMsgs);
                    RequestDispatcher view = request.getRequestDispatcher("/error.jsp");
50                  view.forward(request, response);
51                  return;
52              }
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





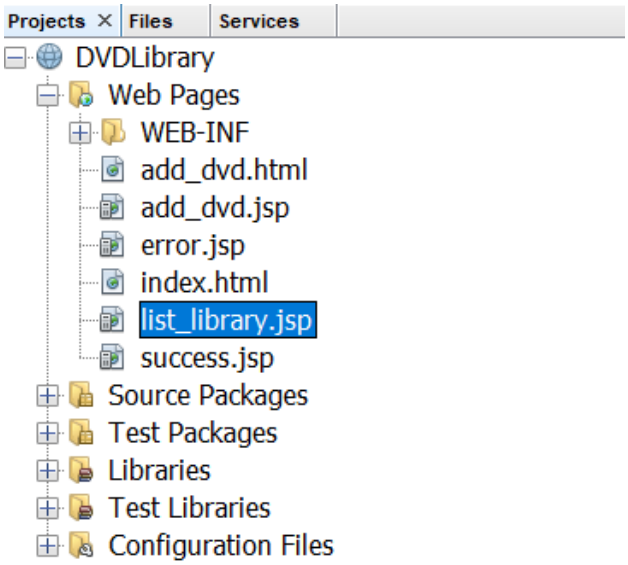




Exercise 3: Converting the List Library View Component to a JSP Page

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.

Source History

```
<%@ page import="java.util.ArrayList" %>
<%@ page import="java.util.Iterator" %>
<%@ page import="com.dvd.model.DVDItem" %>
<%
    // Create a list of DVDs
    List dvds = new ArrayList();
    dvds.add(new DVDItem("Aquaman", "2019", "Sci-Fi"));
    dvds.add(new DVDItem("Robin Hood", "2018", "Action"));
    dvds.add(new DVDItem("The Grinch", "2018", "Comedy"));
%>
<!DOCTYPE html>
<html>
<head>
    <title>List Library</title>
</head>
<body bgcolor='white'>
    You currently have <b><%= dvds.size() %></b> DVDs in your collection:<br>
    <table border='0' cellspacing='0' cellpadding='5'>
        <tr>
            <th>TITLE</th>
            <th>YEAR</th>
            <th>GENRE</th>
        </tr>
        <%
            Iterator it = dvds.iterator();
            while (it.hasNext()) {
                DVDItem item = (DVDItem) it.next();
        %>
        <tr>
            <td><%= item.getTitle() %></td>
            <td><%= item.getYear() %></td>
            <td><%= item.getGenre() %></td>
        </tr>
        <%
            }
        %>
    </table>
    End of list...
</body>
</html>
```

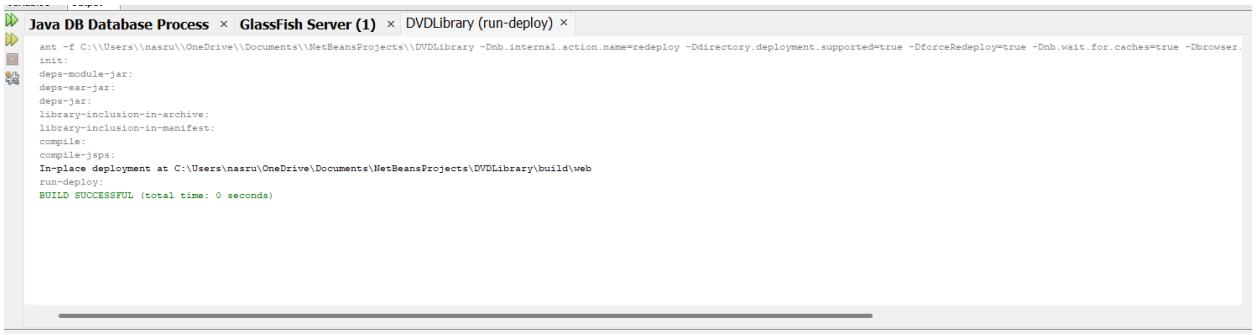
3. Modify the index.html home page to change the link to the new List Library JSP page.

Source History

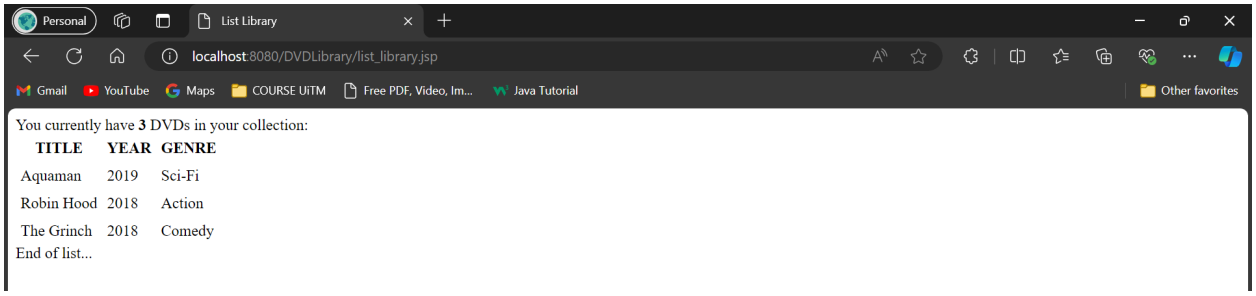
```
<!DOCTYPE html>
<!--
To change this license header, choose License Headers in Project Properties.
To change this template file, choose Tools | Templates
and open the template in the editor.
-->
<html>
    <head>
        <title></title>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">

    </head>
    <body>
        <h1>DVD Library Application</h1>

        <ul>

            <li><a href='list_library.jsp'>Display my DVD library</a></li>
            <li><a href='add_dvd.jsp'>Add a DVD to my collection</a></li>
        </ul>



    </body>
</html>
```

Task 2: Remove the Old Servlet Code (optional)

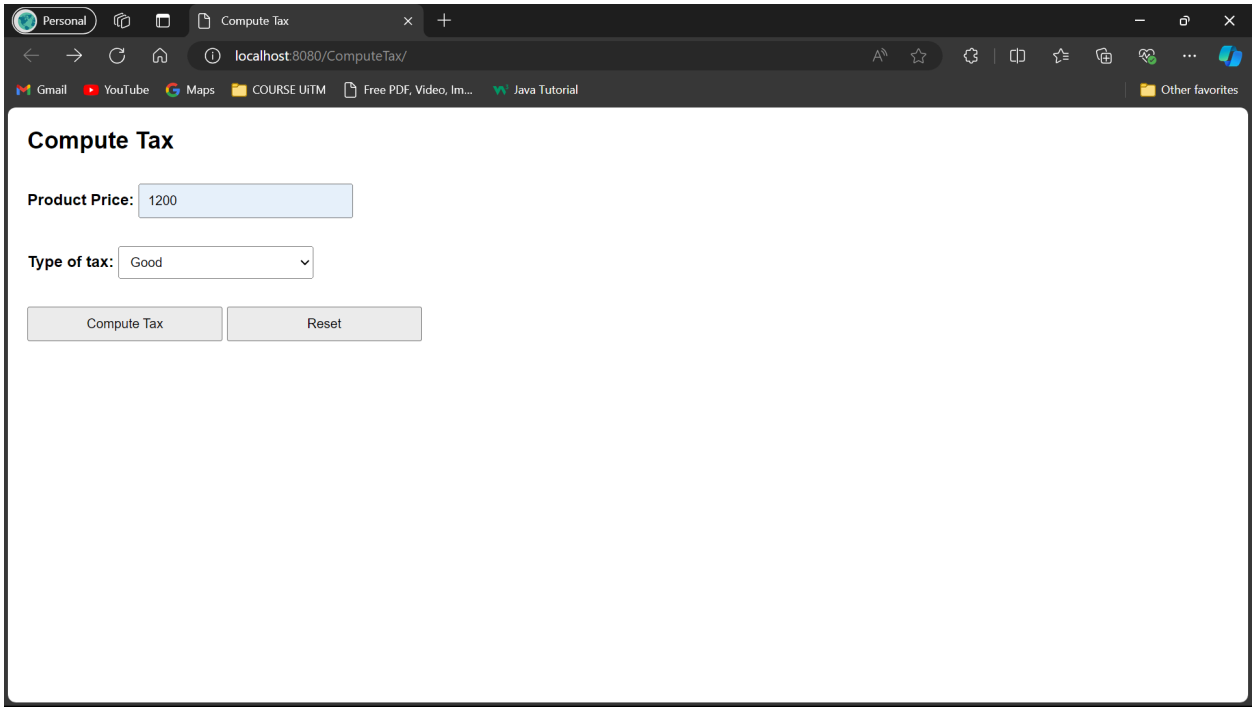
Task 3: Deploy the Web Application

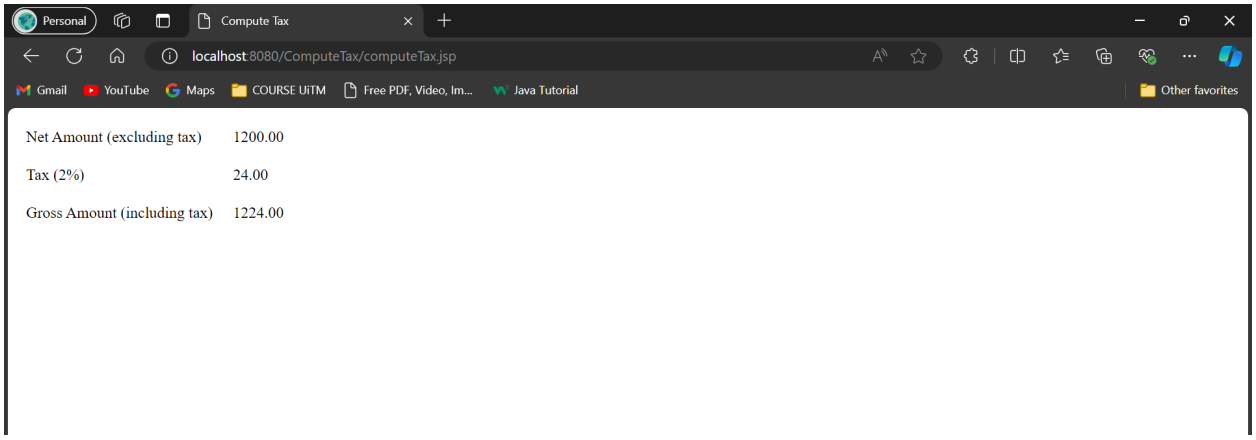


Task 4: Test the Web Application



Post Lab Exercise





LAB 6: Database Programming in Servlets

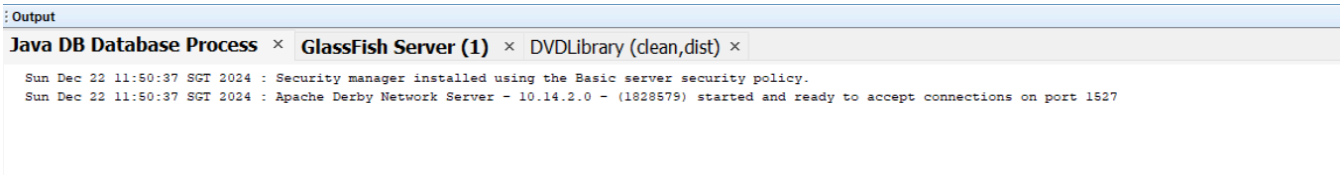
Lab Activities

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:

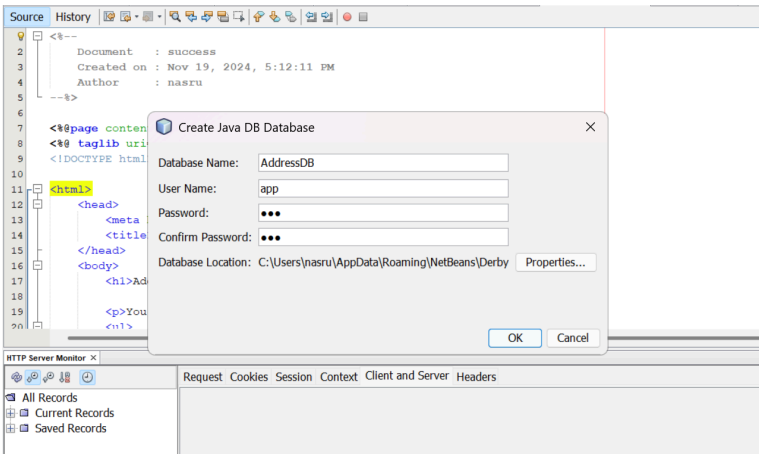
a. In the Services window, right-click the Java DB node and choose Start Server. Note the following output in the Output window, indicating that the server has started:



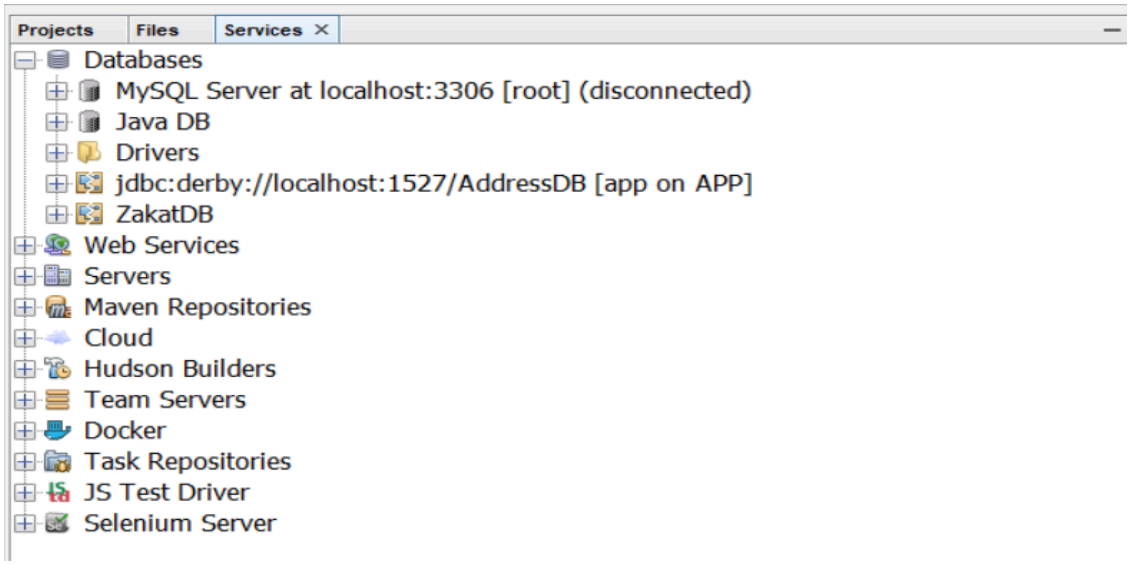
b. Right-click the Java DB node and choose Create Database to open the Create Java DB Database dialog.

c. Type AddressDB for the Database Name.

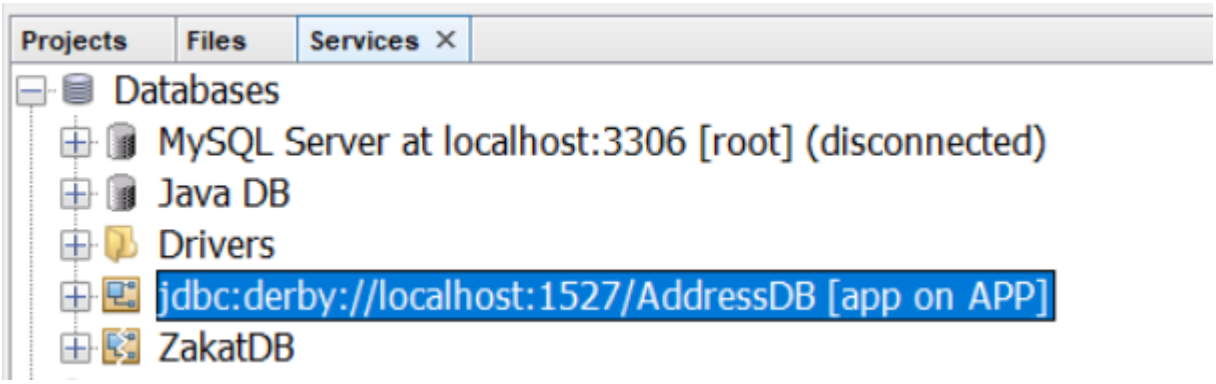
d. Type app for the User Name and Password. Click OK.



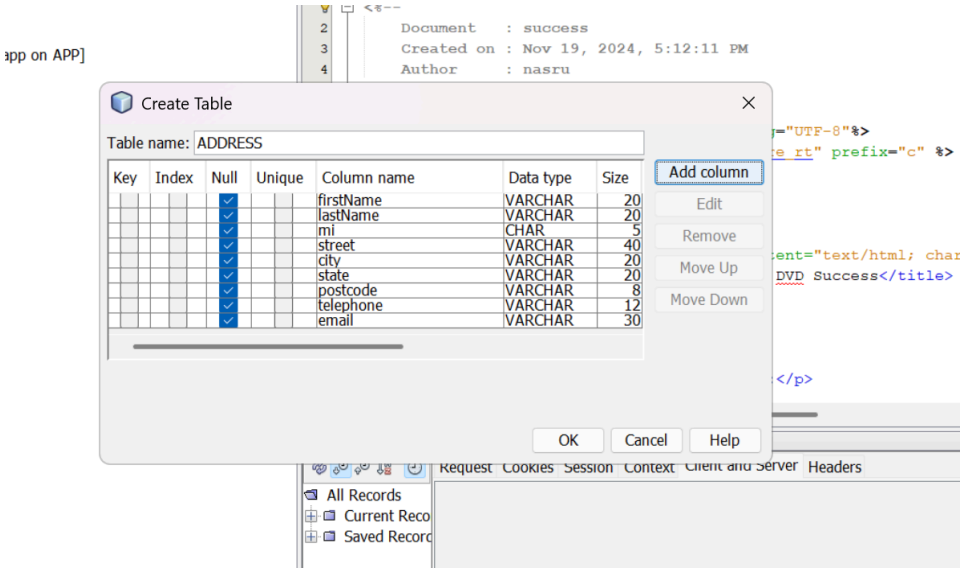
After you create the database, if you expand the Databases node in the Services window you can see that the IDE created a database connection and that the database was added to the list under the Java DB node.



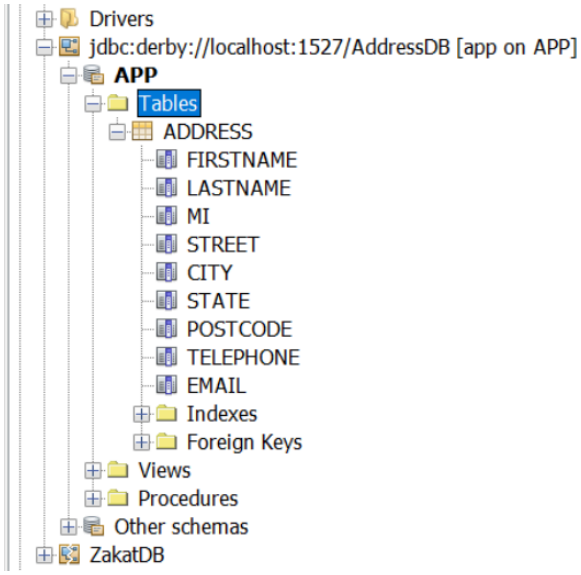
Task 2: Connecting to the Database



Task 3) Creating Tables

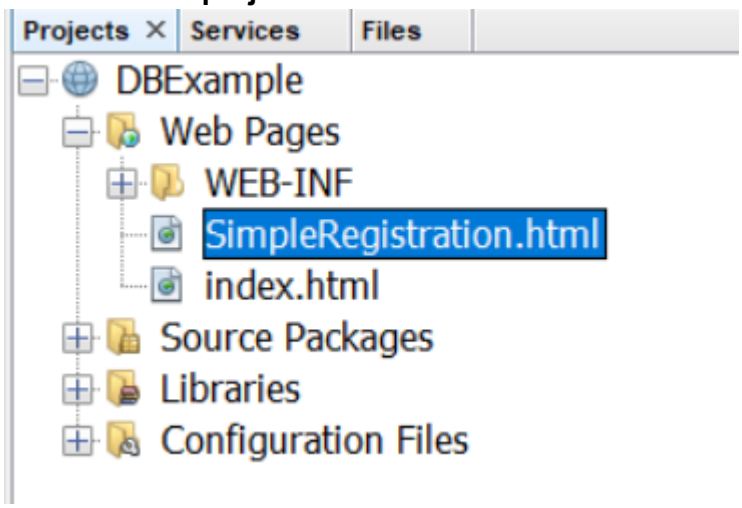


f. When you are sure that your Create Table dialog contains the same specifications as those shown above, click OK. The IDE generates the ADDRESS table in the database, and you can see a new ADDRESS table node ( ) display under the Tables node.



2. Creating web components.

Task 1: Create a web project



Task 2: Create an HTML file

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

index.html | SimpleRegistration.html

Source | History

```
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">
            <p>Last Name <font color="#FF0000">*</font>
17              <input type="text" name="lastName"> 
                First Name <font color="#FF0000">*</font>
19              <input type="text" name="firstName"> 
20              MI <input type="text" name="mi" size="3">
21          </p>
22          <p>Telephone
23              <input type="text" name="telephone" size="20"> 
24              Email
25              <input type="text" name="email" size="30"> 
26          </p>
27          <p>Street <input type="text" name="street" size="40"></p>
28          <p>
29              City <input type="text" name="city" size="20"> 
30              Postcode <input type="text" name="postcode" size="6">
31          </p>
32          <p>     State
33              <select size="1" name="state">
34                  <option value="Selangor">Selangor</option>
35                  <option value="Perak">Perak</option>
36                  <option value="Kedah">Kedah</option>
37                  <option value="Melaka">Melaka</option>
38                  <option value="Johor">Johor</option>
39                  <option value="Negeri Sembilan">Negeri Sembilan</option>
40              </select> 
41          </p><br>
```

Task 3: Create a Servlet

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

index.html | SimpleRegistration.html | SimpleRegistration.java

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 | SimpleRegistration.html | SimpleRegistration.java

Source | History

```
27     @Override
       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");
           } catch (Exception ex) {
51             out.println("Error: "+ex.getMessage());
52         } finally {
53             out.close();
54         }
55     }
56
```

index.html × SimpleRegistration.html × SimpleRegistration.java ×

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
                } catch (Exception ex) {
                    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

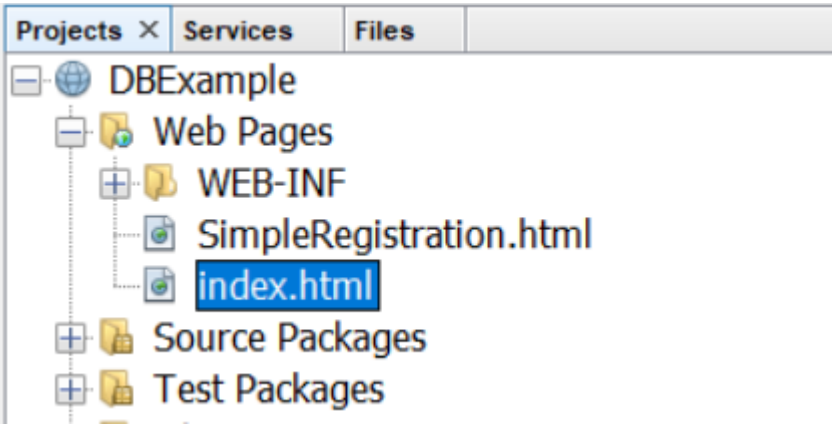
index.html × SimpleRegistration.html × SimpleRegistration.java × web.xml ×

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

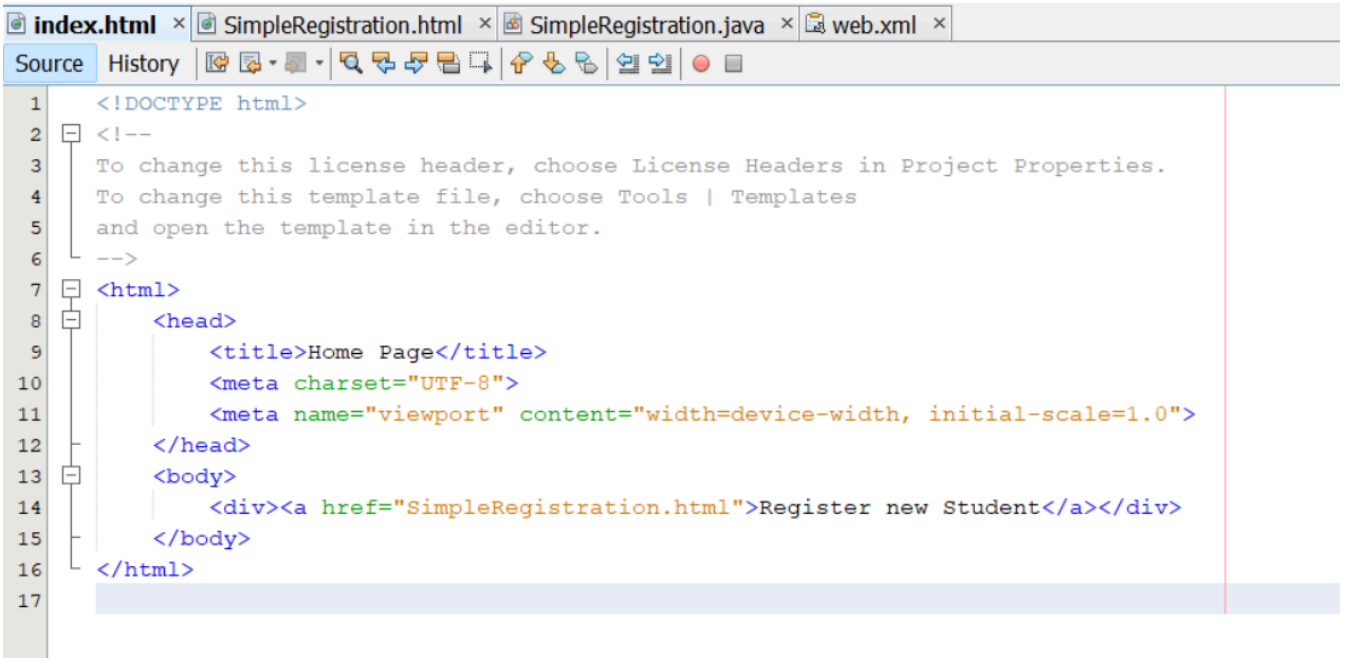
a. Create an HTML file with the following characteristic:



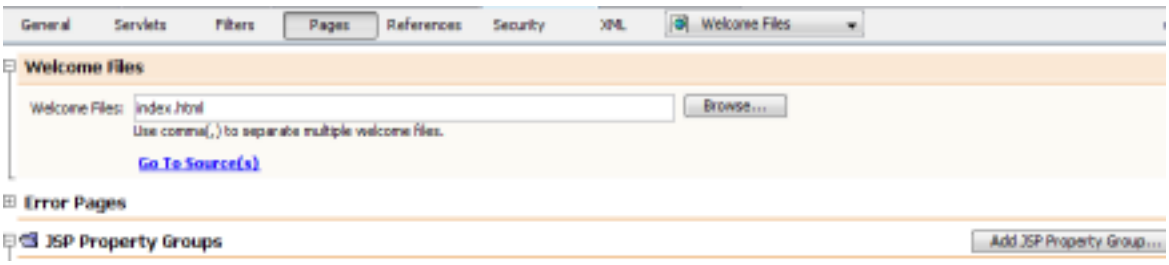
b. Edit the index.html file so that it displays the following text.

Register new student

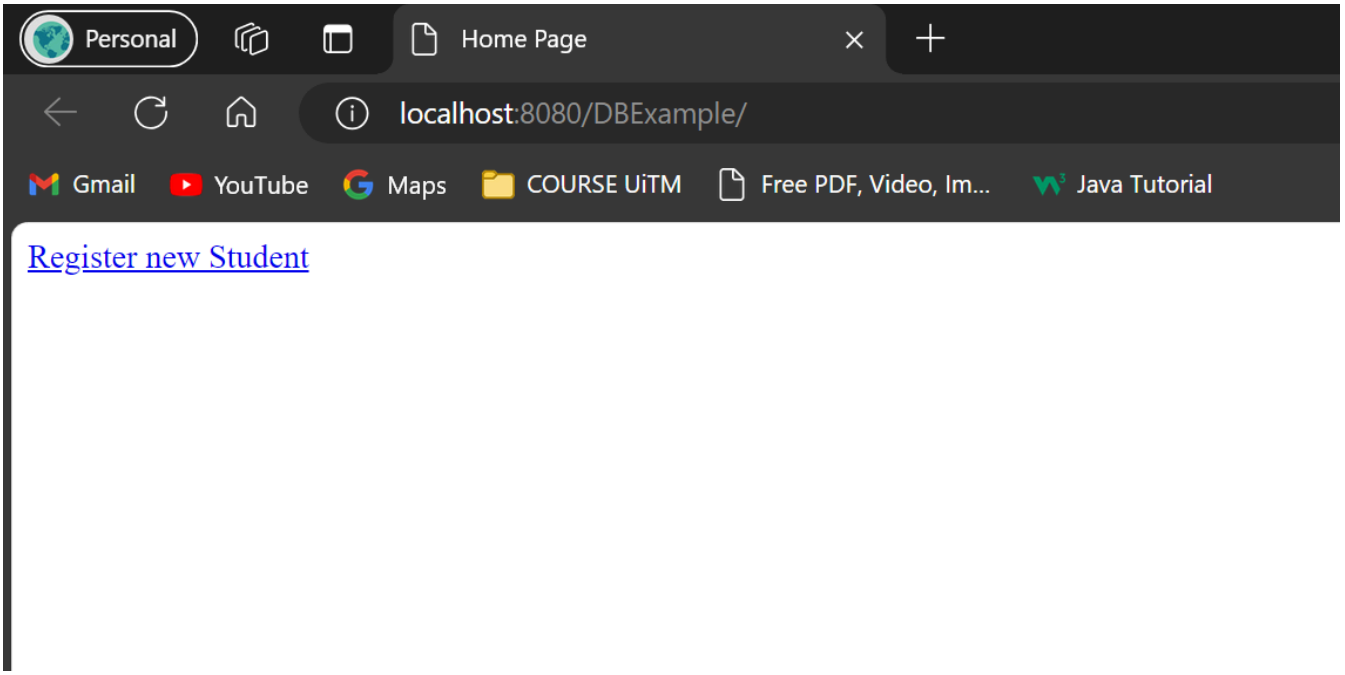
Edit the index.html file to add a link to the SimpleRegistration.html page.

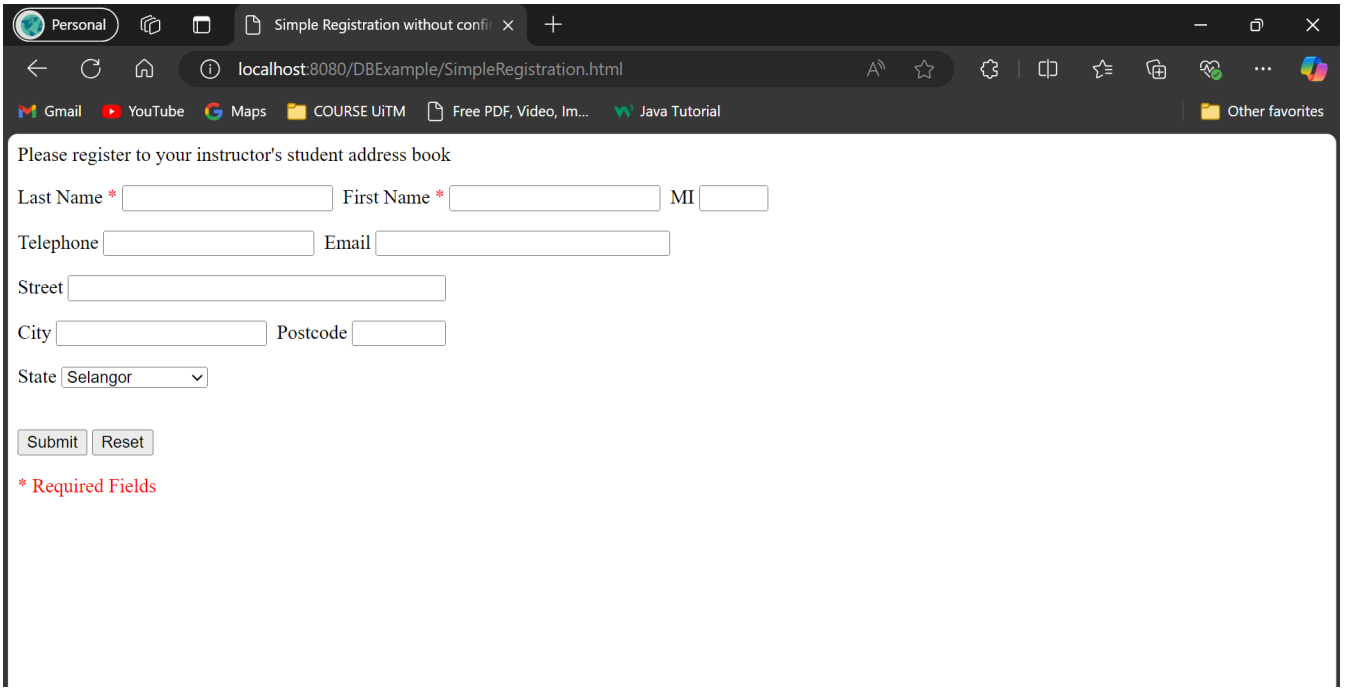


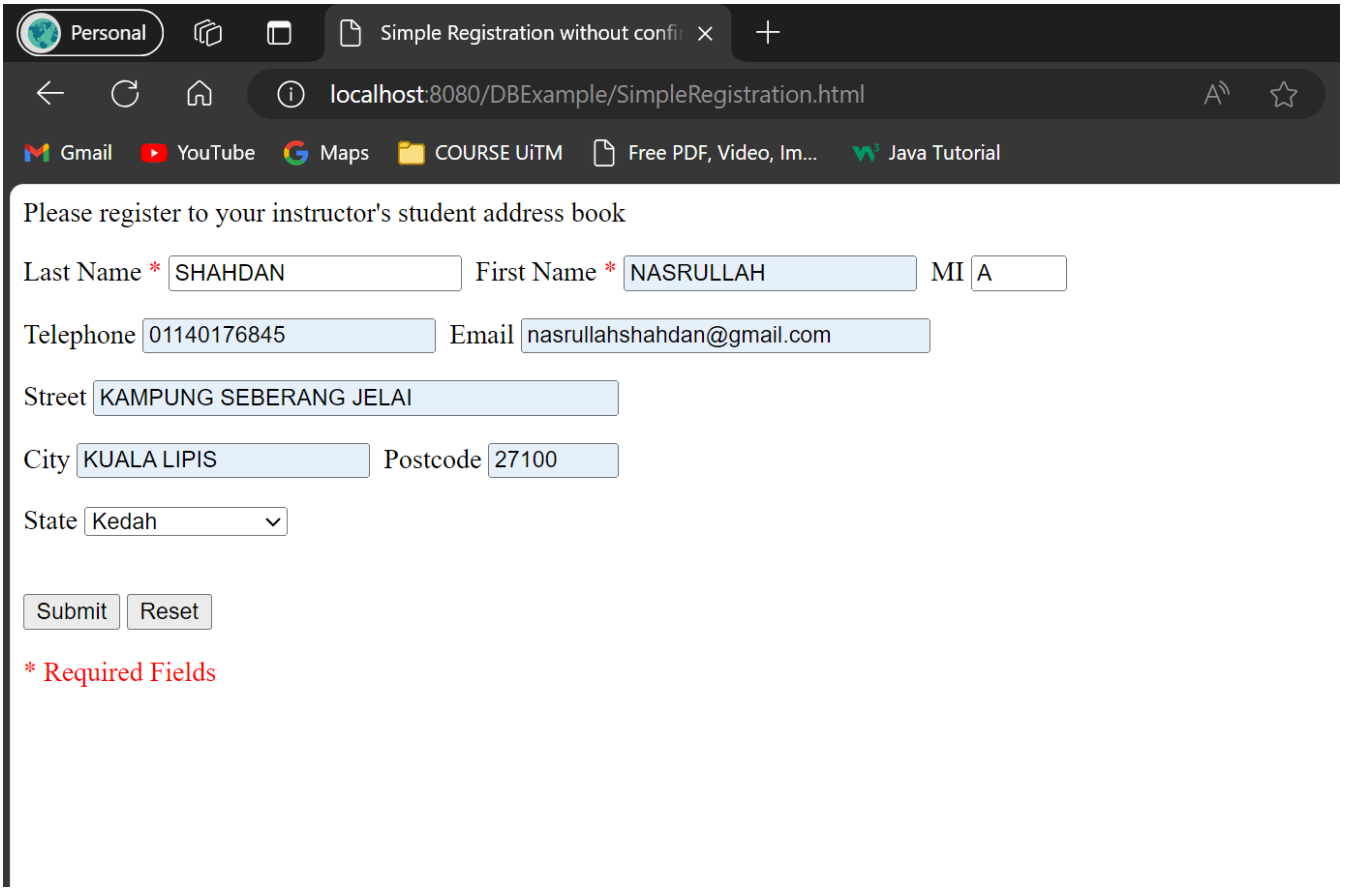
c. Open the web.xml deployment descriptor and set index.html as the welcome file.

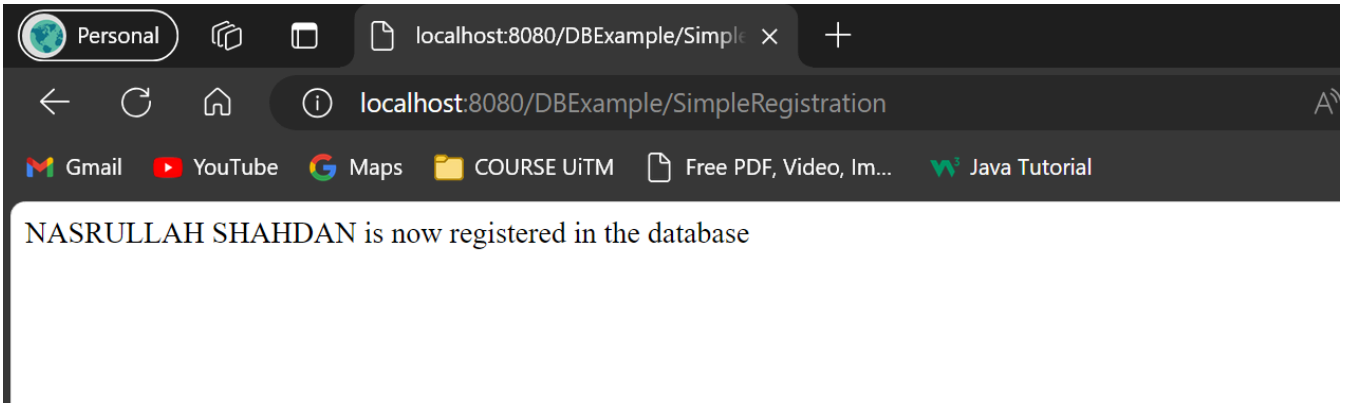


Task 3: Deploy the web application






Personal
Simple Registration without confi
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
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