LAB 5: Session Tracking and Database Operations in Servlets

Lab Activities

You are required to modify the Music Library web application by replacing the ArrayList with a database. The web application allows users to store, retrieve and update music records. The web application implements session tracking to ensure only authorized users can add music records in the database.

1. Create a database and a table in Java Derby Task

1: Starting the Server and Creating a Database

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

b. Type MusicLibraryDB for the Database Name.

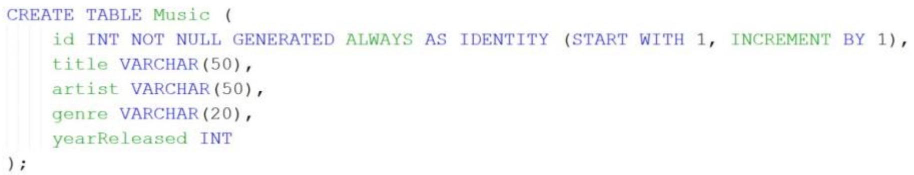
c. Type app for the UserName and Password. Click OK.

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: Creating Table

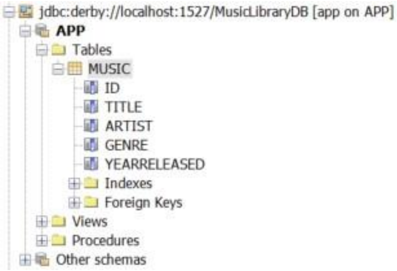
The MusicLibraryDB database that you just created is currently empty. It does not yet contain any tables or data. In this lab, you will use the Execute Command to create table.

a. In the SQL Command Editor, execute the following SQL command to create table with autoincrement id column.

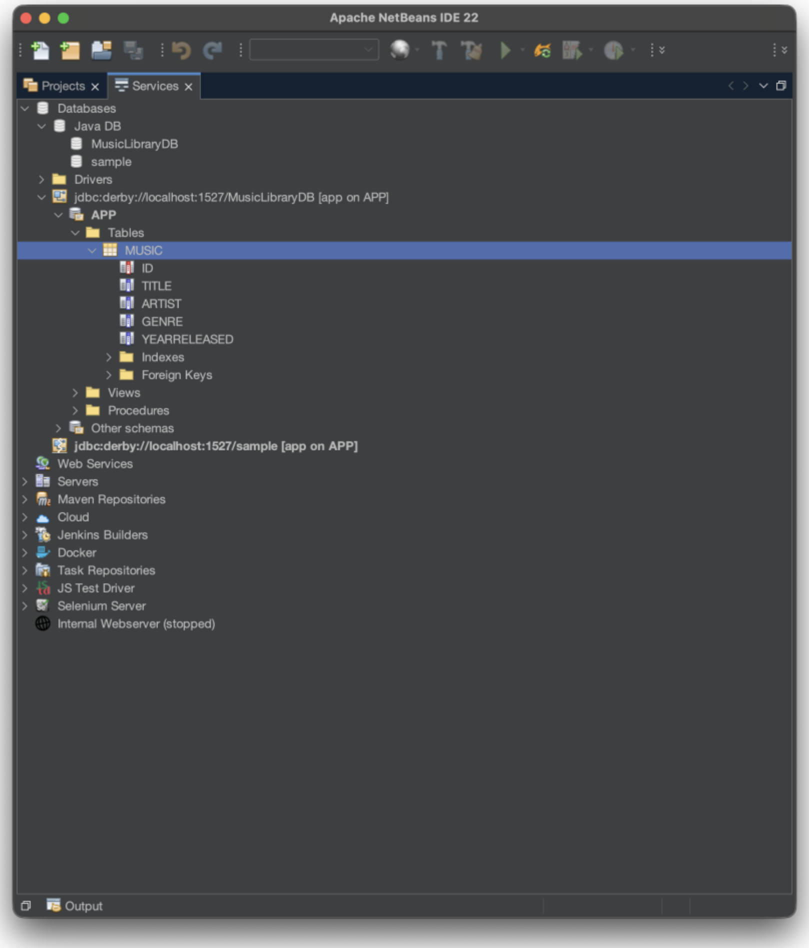


```
CREATE TABLE Music (
    id INT NOT NULL GENERATED ALWAYS AS IDENTITY (START WITH 1, INCREMENT BY 1),
    title VARCHAR(50),
    artist VARCHAR(50),
    genre VARCHAR(20),
    yearReleased INT
);
```

b. Go to the Services window, right-click the Tables node under your database, and choose Refresh. Expand the Tables node to ensure the new Music table has been created successfully.



```
jdbc:derby://localhost:1527/MusicLibraryDB [app on APP]
  APP
    Tables
      MUSIC
        ID
        TITLE
        ARTIST
        GENRE
        YEARRELEASED
        Indexes
        Foreign Keys
    Views
    Procedures
  Other schemas
```



Apache NetBeans IDE 22

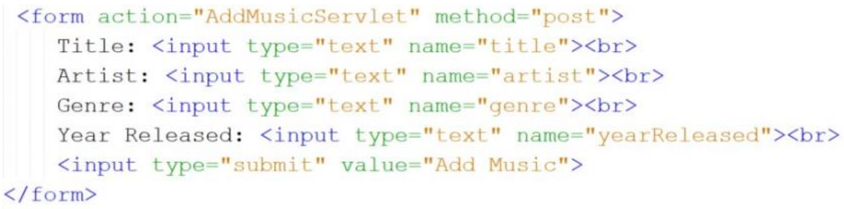
```
Databases
  Java DB
    MusicLibraryDB
    sample
  Drivers
  jdbc:derby://localhost:1527/MusicLibraryDB [app on APP]
    APP
      Tables
        MUSIC
          ID
          TITLE
          ARTIST
          GENRE
          YEARRELEASED
          Indexes
          Foreign Keys
      Views
      Procedures
    Other schemas
  jdbc:derby://localhost:1527/sample [app on APP]
Web Services
Servers
Maven Repositories
Cloud
Jenkins Builders
Docker
Task Repositories
JS Test Driver
Selenium Server
Internal Webserver (stopped)
```

2. Creating web components for the MusicLibrary project

Task 1: Create an HTML form

a. Create an HTML file `add_music.html` for collecting the data and sending it to the database.

b. The form will send data via POST to the `AddMusicServlet` as follows:

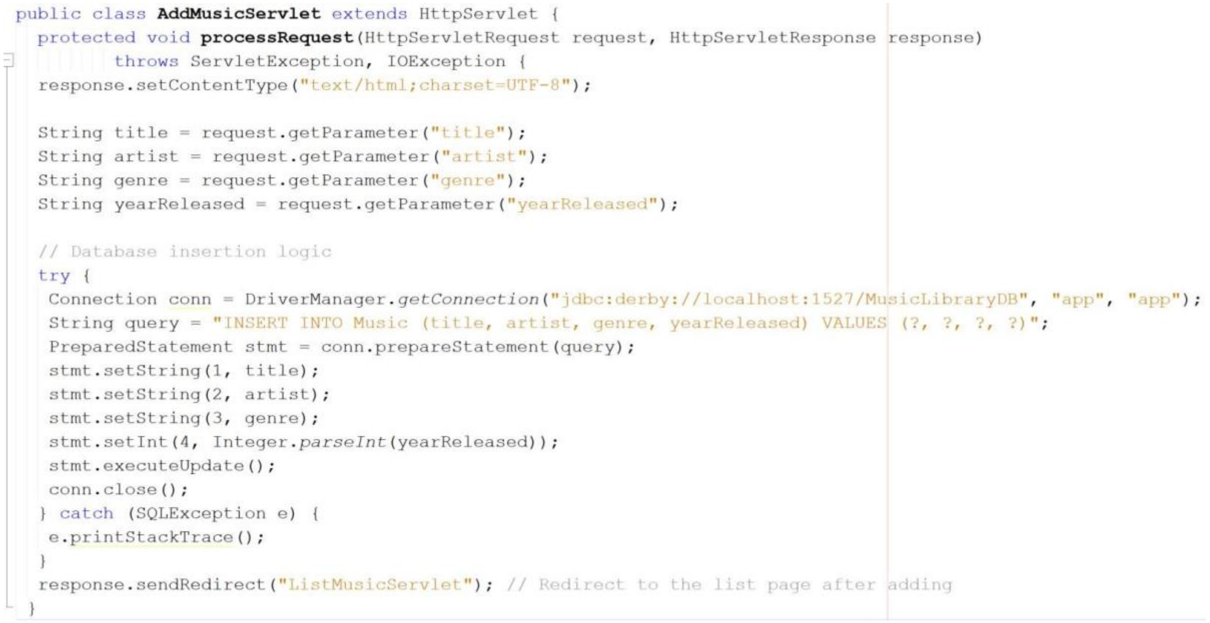


```
<form action="AddMusicServlet" method="post">
    Title: <input type="text" name="title"><br>
    Artist: <input type="text" name="artist"><br>
    Genre: <input type="text" name="genre"><br>
    Year Released: <input type="text" name="yearReleased"><br>
    <input type="submit" value="Add Music">
</form>
```

Task 2: Create a Servlet for adding music items in the database

a. Create a Java Servlet with the following characteristic:
Class name: AddMusicServlet
Project : MusicLibrary
Location:Source Package
Package : default package
Servlet name: AddMusicServlet
URL Pattern : /AddMusicServlet

b. Add the following code in the AddMusicServlet servlet.

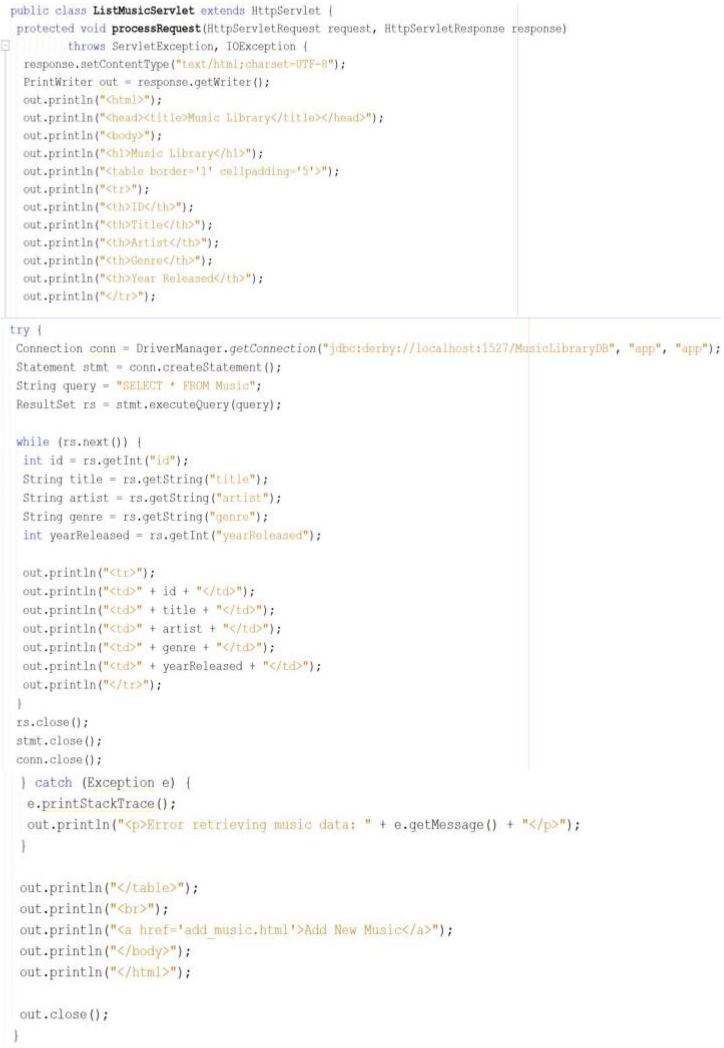


```
public class AddMusicServlet extends HttpServlet {
  protected void processRequest(HttpServletRequest request, HttpServletResponse response)
          throws ServletException, IOException {
  response.setContentType("text/html;charset=UTF-8");

  String title = request.getParameter("title");
  String artist = request.getParameter("artist");
  String genre = request.getParameter("genre");
  String yearReleased = request.getParameter("yearReleased");

  // Database insertion logic
  try {
   Connection conn = DriverManager.getConnection("jdbc:derby://localhost:1527/MusicLibraryDB", "app", "app");
   String query = "INSERT INTO Music (title, artist, genre, yearReleased) VALUES (?, ?, ?, ?)";
   PreparedStatement stmt = conn.prepareStatement(query);
   stmt.setString(1, title);
   stmt.setString(2, artist);
   stmt.setString(3, genre);
   stmt.setInt(4, Integer.parseInt(yearReleased));
   stmt.executeUpdate();
   conn.close();
  } catch (SQLException e) {
   e.printStackTrace();
  }
  response.sendRedirect("ListMusicServlet"); // Redirect to the list page after adding
 }
```

Task 3: Create a Servlet for listing music items in the database

a. Create a Java Servlet with the following characteristic:
Class name: ListMusicServlet
Project : MusicLibrary
Location:Source Package
Package : default package
Servlet name: ListMusicServlet
URL Pattern : /ListMusicServlet

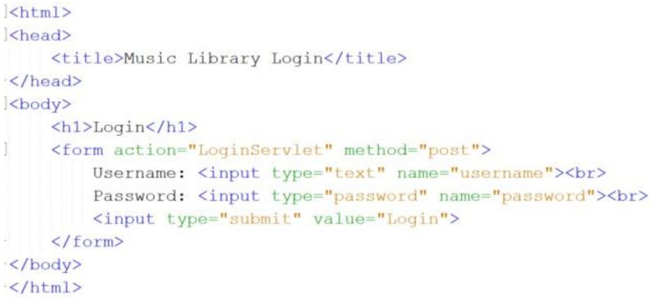
b. Add the following code in the ListMusicServlet servlet.



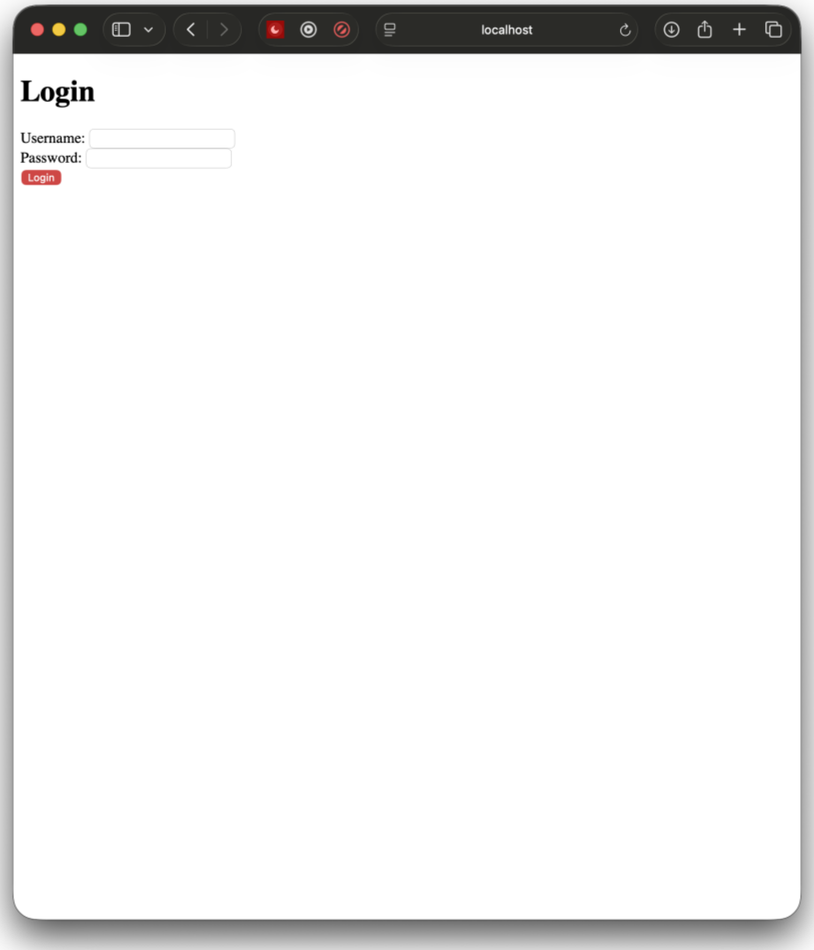
```
public class ListMusicServlet extends HttpServlet {
 protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
  response.setContentType("text/html;charset=UTF-8");
  PrintWriter out = response.getWriter();
  out.println("<html>");
  out.println("<head><title>Music Library</title></head>");
  out.println("<body>");
  out.println("<h1>Music Library</h1>");
  out.println("<table border='1' cellpadding='5'>");
  out.println("<tr>");
  out.println("<th>ID</th>");
  out.println("<th>Title</th>");
  out.println("<th>Artist</th>");
  out.println("<th>Genre</th>");
  out.println("<th>Year Released</th>");
  out.println("</tr>");

try {
 Connection conn = DriverManager.getConnection("jdbc:derby://localhost:1527/MusicLibraryDB", "app", "app");
 Statement stmt = conn.createStatement();
 String query = "SELECT * FROM Music";
 ResultSet rs = stmt.executeQuery(query);

 while (rs.next()) {
  int id = rs.getInt("id");
  String title = rs.getString("title");
  String artist = rs.getString("artist");
  String genre = rs.getString("genre");
  int yearReleased = rs.getInt("yearReleased");

  out.println("<tr>");
  out.println("<td>" + id + "</td>");
  out.println("<td>" + title + "</td>");
  out.println("<td>" + artist + "</td>");
  out.println("<td>" + genre + "</td>");
  out.println("<td>" + yearReleased + "</td>");
  out.println("</tr>");
 }
 rs.close();
 stmt.close();
 conn.close();
 } catch (Exception e) {
  e.printStackTrace();
  out.println("<p>Error retrieving music data: " + e.getMessage() + "</p>");
 }

 out.println("</table>");
 out.println("<br>");
 out.println("<a href='add_music.html'>Add New Music</a>");
 out.println("</body>");
 out.println("</html>");

 out.close();
}
```

3. Implementing session tracking for the MusicLibrary project Task 1 : Create an html for Login form (login.html)

a. The html form submits the username and password to LoginServlet using the post method as the following code:



```
<html>
<head>
    <title>Music Library Login</title>
</head>
<body>
    <h1>Login</h1>
    <form action="LoginServlet" method="post">
        Username: <input type="text" name="username"><br>
        Password: <input type="password" name="password"><br>
        <input type="submit" value="Login">
    </form>
</body>
</html>
```



Login

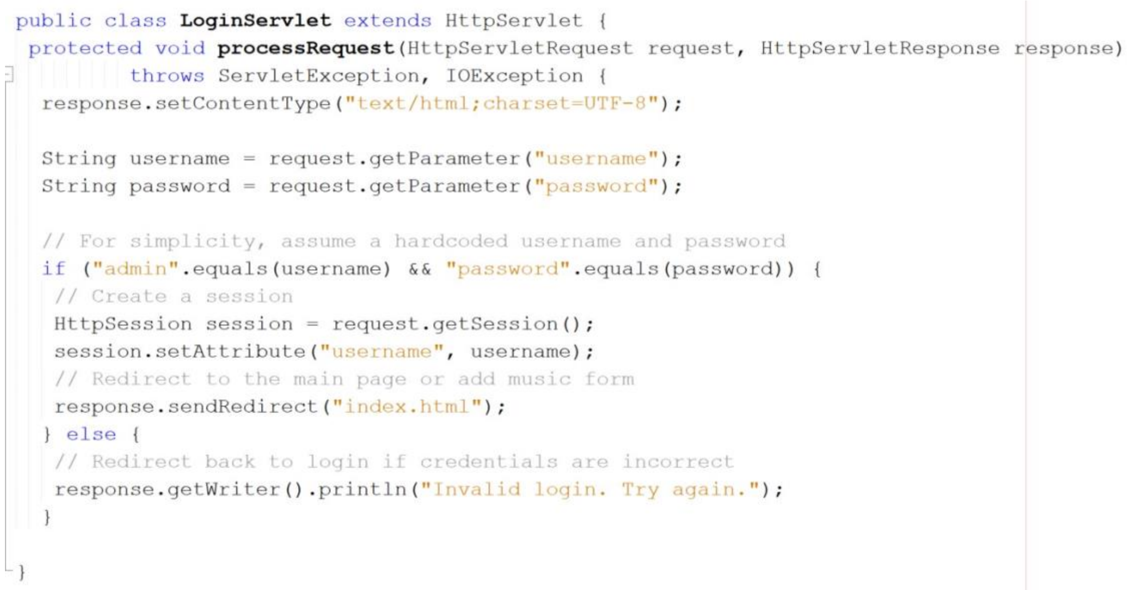
Username:

Password:

Task 2 : Create a LoginServlet

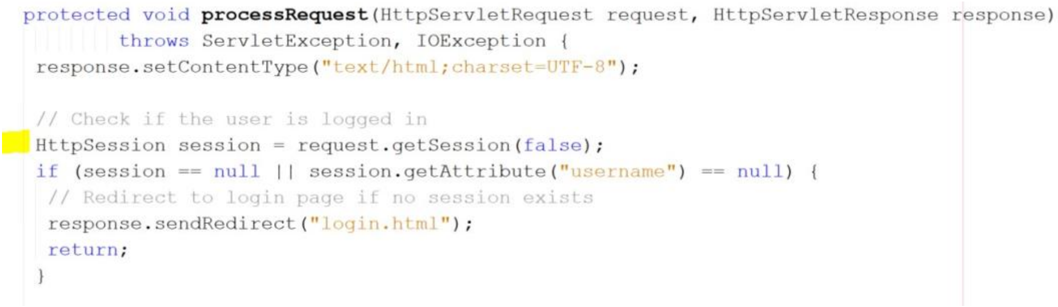
a. Create a Java Servlet with the following characteristic:
Class name: LoginServlet
Project : MusicLibrary
Location:Source Package
Package : default package
Servlet name: LoginServlet
URL Pattern : /LoginServlet

b. Add the following code in the LoginServlet servlet.



```
public class LoginServlet extends HttpServlet {
 protected void processRequest(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException {
  response.setContentType("text/html;charset=UTF-8");

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

  // For simplicity, assume a hardcoded username and password
  if ("admin".equals(username) && "password".equals(password)) {
   // Create a session
   HttpSession session = request.getSession();
   session.setAttribute("username", username);
   // Redirect to the main page or add music form
   response.sendRedirect("index.html");
  } else {
   // Redirect back to login if credentials are incorrect
   response.getWriter().println("Invalid login. Try again.");
  }

}
```

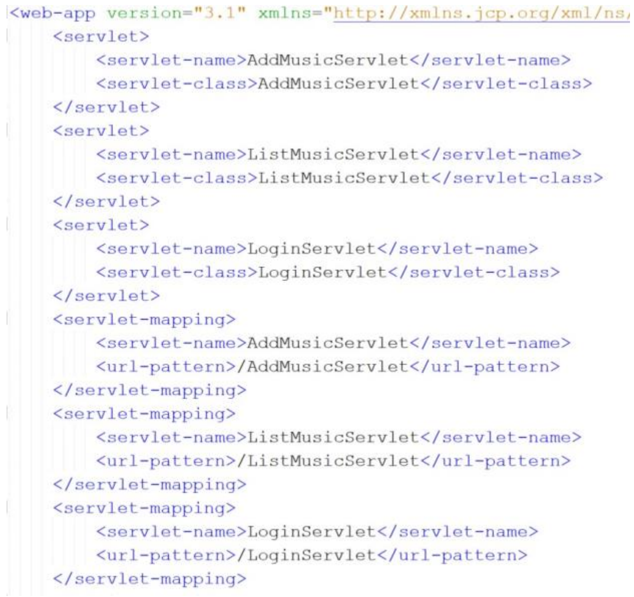
c. Modify AddMusicServlet to check for a session. Add the following code to check the session.



```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
 response.setContentType("text/html;charset=UTF-8");

 // Check if the user is logged in
 HttpSession session = request.getSession(false);
 if (session == null || session.getAttribute("username") == null) {
  // Redirect to login page if no session exists
  response.sendRedirect("login.html");
  return;
 }
```

4. Configuring the Web application Task 1 : Verify the view servlet configuration

a. Open the web.xml deployment descriptor in the Web Pages – Web INF folder.
b. View the definition and the mapping for the two servlets.

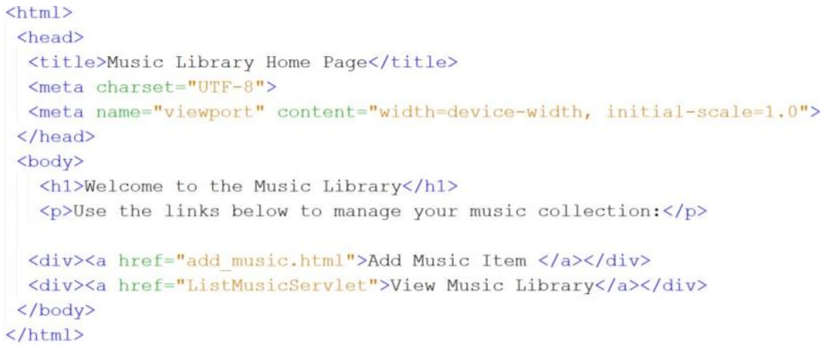


```
<web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/
    <servlet>
        <servlet-name>AddMusicServlet</servlet-name>
        <servlet-class>AddMusicServlet</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>ListMusicServlet</servlet-name>
        <servlet-class>ListMusicServlet</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>LoginServlet</servlet-name>
        <servlet-class>LoginServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>AddMusicServlet</servlet-name>
        <url-pattern>/AddMusicServlet</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>ListMusicServlet</servlet-name>
        <url-pattern>/ListMusicServlet</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>LoginServlet</servlet-name>
        <url-pattern>/LoginServlet</url-pattern>
    </servlet-mapping>
```

Task 2: Create the homepage

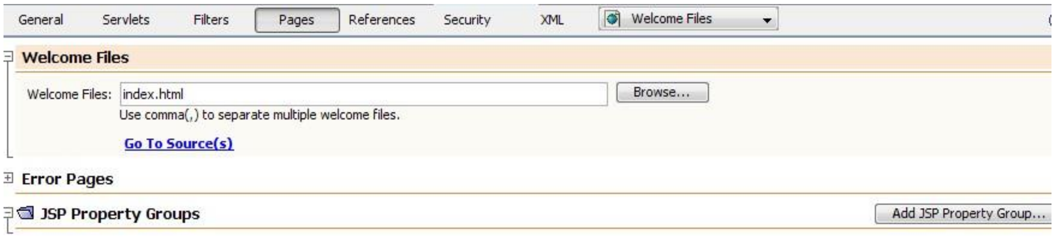
a. Create an HTML file with the following characteristic:
Project : MusicLibrary
HTML file name: index
Location:Web Pages

b. Edit the index.html file as follows.



```
<html>
 <head>
  <title>Music Library Home Page</title>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
 </head>
 <body>
   <h1>Welcome to the Music Library</h1>
   <p>Use the links below to manage your music collection:</p>

  <div><a href="add_music.html">Add Music Item </a></div>
  <div><a href="ListMusicServlet">View Music Library</a></div>
 </body>
</html>
```

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



Welcome Files

Welcome Files: index.html

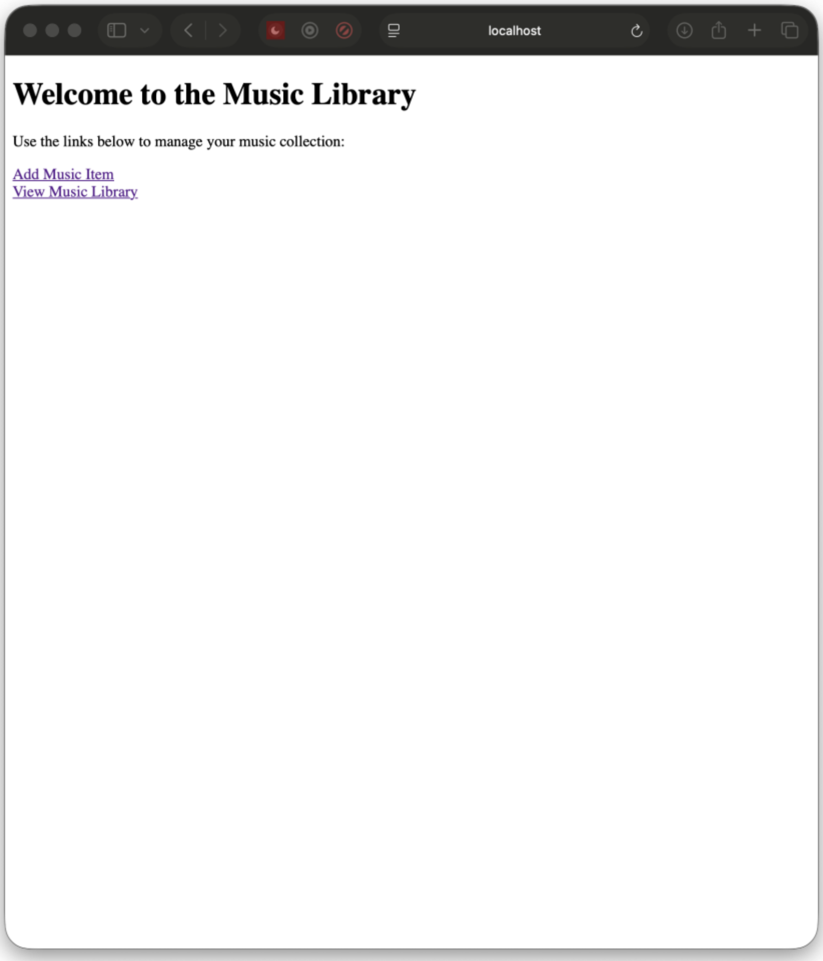
Use comma(,) to separate multiple welcome files.

Error Pages

JSP Property Groups

Task 3: Deploy the web application

a. Build the MusicLibrary web application. Correct any errors you encounter
b. Deploy the MusicLibrary web Application
c. Run the web application project



Welcome to the Music Library

Use the links below to manage your music collection:

Add Music Item
View Music Library

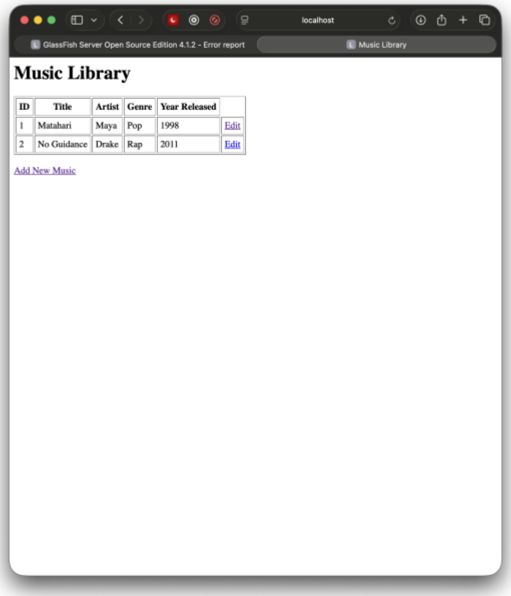
Postlab Exercise

You need to implement the functionality to edit exisiting music records in a database using HTML forms and Servlets. The user will be able to edit the details of a music record (title, artist, genre, and year released) in the database. Perform the following tasks:

1. Create an HTML form (`edit_music.html`) that allows the user to edit the details of a selected music record. The form should include fields for the title, artist, genre, and year released, prefilled with the current values of the selected record. When the user submits the form, it should send the data to the `EditMusicServlet` for processing.

2. Write the `EditMusicServlet` that performs the following tasks:
   a. Check for an active session (similar to `AddMusicServlet`)
   b. Retrieve the form data (title, artist, genre, year released, and id).
   c. Update the database:
   d. Redirect the user to the list of music records (`ListMusicServlet`) after the update is successful.

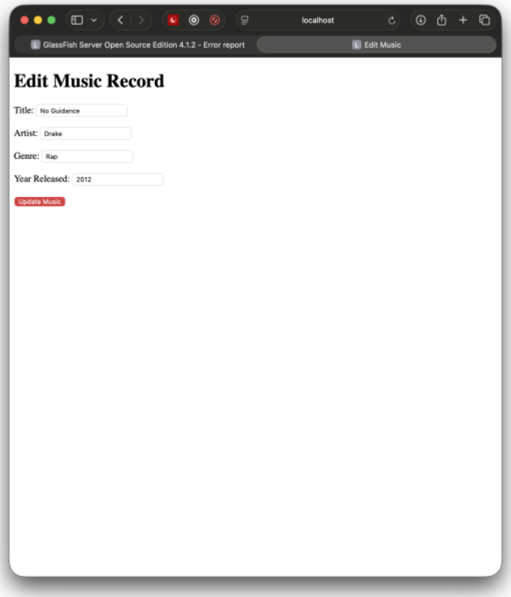
3. Modify the `ListMusicServlet` to include an **"Edit"** link next to each music record. The **"Edit"** link should direct the user to the `edit_music.html` form, pre-filled with the current details of the selected music record.



Music Library

ID	Title	Artist	Genre	Year Released	
1	Matahari	Maya	Pop	1998	Edit
2	No Guidance	Drake	Rap	2011	Edit

Add New Music



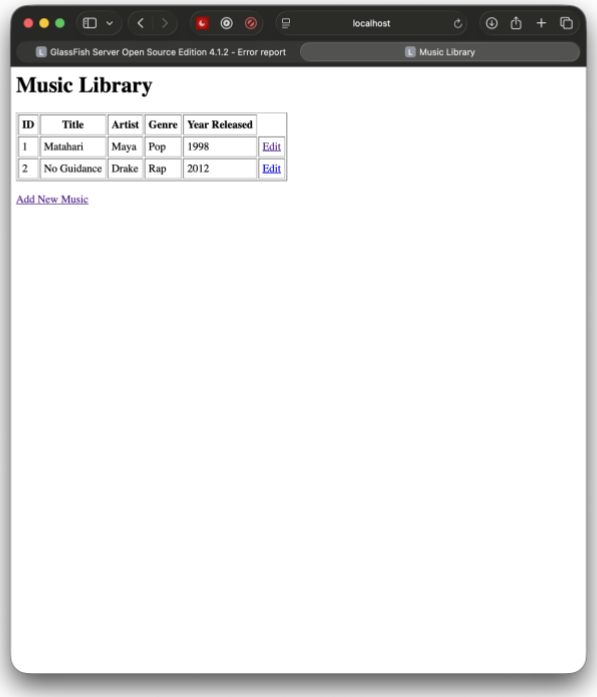
Edit Music Record

Title: No Guidance

Artist: Drake

Genre: Rap

Year Released: 2012



Music Library

ID	Title	Artist	Genre	Year Released	
1	Matahari	Maya	Pop	1998	Edit
2	No Guidance	Drake	Rap	2012	Edit

Add New Music

edit_music.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Edit Music</title>
</head>
<body>
    <h1>Edit Music Record</h1>

    <form action="EditMusicServlet" method="post">

        <input type="hidden" name="id">

        Title:
        <input type="text" name="title"><br><br>

        Artist:
        <input type="text" name="artist"><br><br>

        Genre:
        <input type="text" name="genre"><br><br>

        Year Released:
        <input type="text" name="year"><br><br>

        <input type="submit" value="Update Music">
    </form>

<script>
    const urlParams = new URLSearchParams(window.location.search);

    document.querySelector('input[name="id"]').value = urlParams.get("id") || "";
    document.querySelector('input[name="title"]').value = urlParams.get("title")
|| "";
    document.querySelector('input[name="artist"]').value =
urlParams.get("artist") || "";
    document.querySelector('input[name="genre"]').value = urlParams.get("genre")
|| "";
    document.querySelector('input[name="year"]').value = urlParams.get("year") ||
"";
</script>

</body>
</html>
```

editMusicServlet.java

```
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import javax.servlet.annotation.WebServlet;

@WebServlet("/EditMusicServlet")
public class EditMusicServlet extends HttpServlet {

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse 
response)
            throws ServletException, IOException {

        HttpSession session = request.getSession(false);
        if (session == null) {
            response.sendRedirect("login.html");
            return;
        }

        String id = request.getParameter("id");
        String title = request.getParameter("title");
        String artist = request.getParameter("artist");
        String genre = request.getParameter("genre");
        String year = request.getParameter("year");

        Connection conn = null;
        PreparedStatement stmt = null;

        try {
            Class.forName("org.apache.derby.jdbc.ClientDriver");
            conn = DriverManager.getConnection(
                    "jdbc:derby://localhost:1527/MusicLibraryDB", "app", "app");

            String sql = "UPDATE Music SET title=?, artist=?, genre=?, 
yearReleased=? WHERE id=?";
            stmt = conn.prepareStatement(sql);
            stmt.setString(1, title);
            stmt.setString(2, artist);
            stmt.setString(3, genre);
            stmt.setString(4, year);
            stmt.setInt(5, Integer.parseInt(id));

            stmt.executeUpdate();

            response.sendRedirect("ListMusicServlet");

        } catch (Exception e) {
            throw new ServletException(e);
        }
    }
}
```