National University of Computer & Emerging Sciences, Karachi

Computer Science Department

Fall 2024, Lab Manual - 09

Course Code: SL3001	**Course: Software Development and construction**
Instructor:	**Yasir Arfat**

Lab # 09

Objective

The objective of this lab is to familiarize students with session and cookie management in Java web applications. Students will learn how to use the HttpSession interface to manage user sessions, store attributes, and handle session expiration. They will implement features such as user login/logout systems, personalized dashboards, and session-based authentication

Introduction to Web in Java

Java provides powerful tools and frameworks for building dynamic web applications. At the core, Java web development relies on servlets and JavaServer Pages (JSP) to handle HTTP requests and generate dynamic content. A web server like Apache Tomcat is used to deploy and manage these applications, allowing Java code to interact seamlessly with front-end elements such as HTML, CSS, and JavaScript.

Java web applications follow the client-server model, where the client sends requests through a web browser, and the server processes them to respond with the appropriate data. Java's robust libraries enable developers to manage sessions, cookies, and form handling while ensuring security through built-in features like authentication and encryption. With frameworks such as Spring and Hibernate, Java web development becomes even more efficient for building scalable enterprise applications. This versatility makes Java a preferred choice for ERP systems, e-commerce platforms, and web portals.

Web App in Demo Project

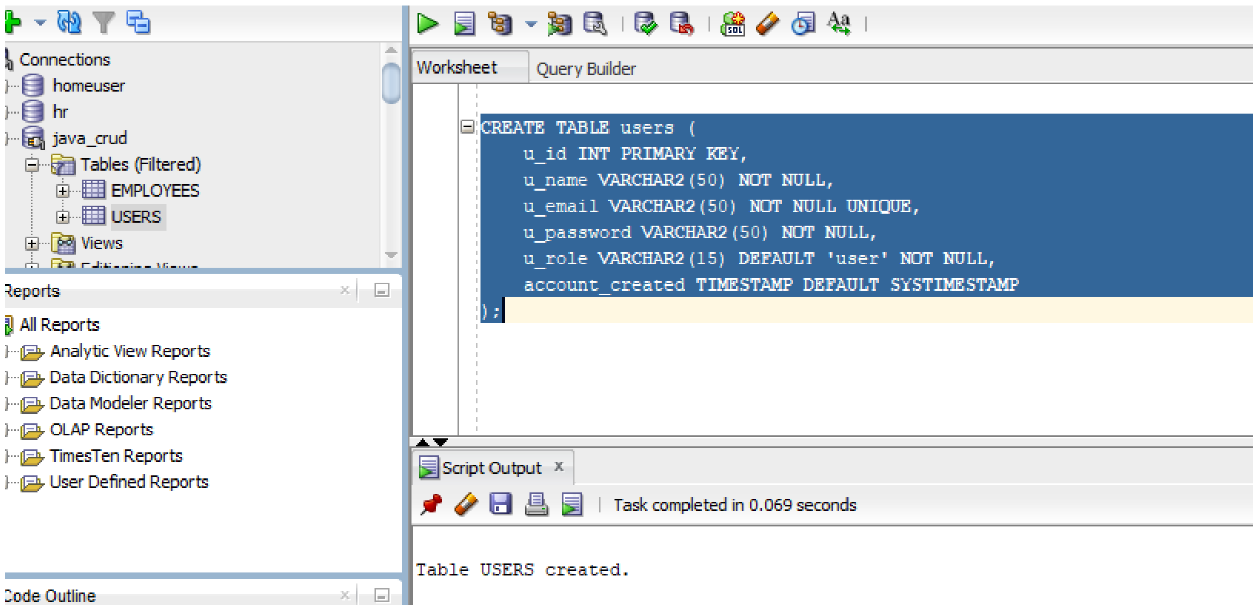
Prerequisites

Before starting, ensure the following tools and configurations are in place:

1. **Software and Tools:**
   - **IntelliJ IDEA**: For developing the web application.
   - **Apache Tomcat 10**: As the web server for deployment.
   - **Oracle SQL Developer**: To manage and interact with the Oracle database.
   - **ojdbc11.jar**: Oracle JDBC driver for database connectivity.
2. **Database Setup:**
   - Create an **Oracle Database connection** in SQL Developer.

We will use the same database we used in last lab for the Demo Project

- Open **Oracle SQL Developer**.
- Connect to your Oracle Database using your credentials.
- Use the following SQL query to create the user table in same Database:



```
CREATE TABLE users (
    u_id INT PRIMARY KEY,
    u_name VARCHAR2(50) NOT NULL,
    u_email VARCHAR2(50) NOT NULL UNIQUE,
    u_password VARCHAR2(50) NOT NULL,
    u_role VARCHAR2(15) DEFAULT 'user' NOT NULL,
    account_created TIMESTAMP DEFAULT SYSTIMESTAMP
);
```

Task completed in 0.069 seconds

```
Table USERS created.
```

CREATE TABLE users (

u_id INT PRIMARY KEY,

```
   u_name VARCHAR2(50) NOT NULL,
   u_email VARCHAR2(50) NOT NULL UNIQUE,
   u_password VARCHAR2(50) NOT NULL,
   u_role VARCHAR2(15) DEFAULT 'user' NOT NULL,
   account_created TIMESTAMP DEFAULT SYSTIMESTAMP
);
```

- **Open IntelliJ** and go to File > New > Project.
- Select **Maven** from the list, click **Next**.
- Enter the following details:
- **Group ID**: com.example
- **Artifact ID**: EmployeeCRUD
- Check **Create from archetype**, and select maven-archetype-webapp.
- Click **Finish**.

Add ojdbc11.jar to Your Project

- Go to `File` > `Project Structure` > **Libraries**.
- Click **+** > Add **Java Library**.
- Select the **ojdbc11.jar** file from your system.
- Click **Apply** and **OK**.

Update pom.xml with Dependencies

- You'll need dependencies for **Servlet API** and **JSP** (since Tomcat requires them) and **ojdbc11.jar**.
- Modify your pom.xml by adding the following dependencies inside the <dependencies> tag:

```
<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.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.example</groupId>
  <artifactId>EmployeeCRUDApp</artifactId>
  <packaging>war</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>EmployeeCRUDApp Maven Webapp</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
```

```
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>

    <!-- Servlet API for Tomcat -->
    <dependency>
      <groupId>jakarta.servlet</groupId>
      <artifactId>jakarta.servlet-api</artifactId>
      <version>5.0.0</version>
      <scope>provided</scope>
    </dependency>

    <!-- JSP support -->
    <dependency>
      <groupId>jakarta.servlet.jsp</groupId>
      <artifactId>jakarta.servlet.jsp-api</artifactId>
      <version>3.0.0</version>
      <scope>provided</scope>
    </dependency>

    <dependency>
      <groupId>com.oracle.database.jdbc</groupId>
      <artifactId>ojdbc11</artifactId>
      <version>21.5.0.0</version> <!-- Use the appropriate version -->
    </dependency>



  </dependencies>
  <build>
    <finalName>EmployeeCRUD</finalName>
  </build>
</project>
```

Create the JDBC Connection Class

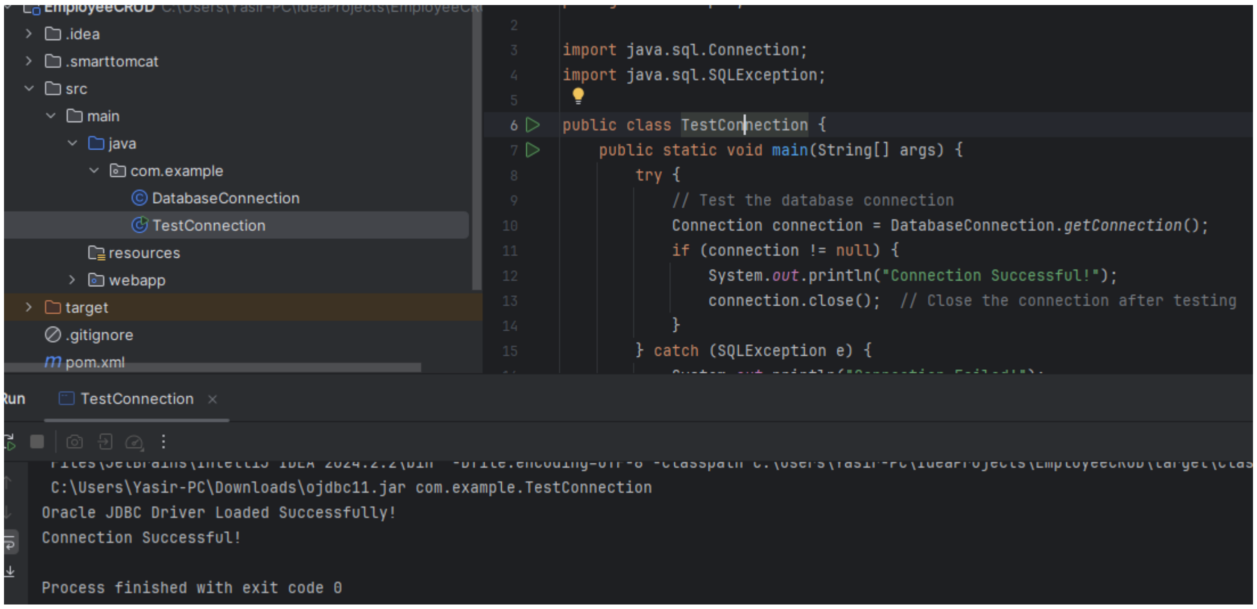
- **Create a new Java class** in IntelliJ:
- Right-click on src/main/java/com/example > New > Java Class.
- Name the class: **DatabaseConnection**
- **Code for DatabaseConnection.java**:

```
package com.example;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DatabaseConnection {
    private static final String URL = "jdbc:oracle:thin:@localhost:1521:xe";
// Adjust as per your setup
    private static final String USER = "java_crud";  // Replace with your
Oracle username
    private static final String PASSWORD = "fast";  // Replace with your
Oracle password

    public static Connection getConnection() throws SQLException {
        try {
            // Load Oracle JDBC Driver
            Class.forName("oracle.jdbc.OracleDriver");
            System.out.println("Oracle JDBC Driver Loaded Successfully!");
        } catch (ClassNotFoundException e) {
            System.out.println("Failed to load Oracle JDBC Driver");
            e.printStackTrace();
        }

        // Establish and return the connection
        return DriverManager.getConnection(URL, USER, PASSWORD);
    }
}
```

- Once you've added this class, try running the following code to **test the connection**:
- In the same Package add another class TestConnection and add the following Code

```
package com.example;

import java.sql.Connection;
import java.sql.SQLException;

public class TestConnection {
    public static void main(String[] args) {
        try {
            // Test the database connection
            Connection connection = DatabaseConnection.getConnection();
            if (connection != null) {
                System.out.println("Connection Successful!");
                connection.close();  // Close the connection after testing
            }
        } catch (SQLException e) {
            System.out.println("Connection Failed!");
            e.printStackTrace();
        }
    }
}
```

- Now run this code to see if you successfully connected to your database or not



```java

import java.sql.Connection;
import java.sql.SQLException;

public class TestConnection {
    public static void main(String[] args) {
        try {
            // Test the database connection
            Connection connection = DatabaseConnection.getConnection();
            if (connection != null) {
                System.out.println("Connection Successful!");
                connection.close();  // Close the connection after testing
            }
        } catch (SQLException e) {
```

```
C:\Users\Yasir-PC\Downloads\ojdbc11.jar com.example.TestConnection
Oracle JDBC Driver Loaded Successfully!
Connection Successful!

Process finished with exit code 0
```

- Now our project is connected to the database we will start working on your project
- Update your index.jsp page with following code for adding the new user

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>User Management App</title>
    <script>
        function showError(message) {
            alert(message);
        }
    </script>
</head>
<body>
<h1>Add User</h1>
<form action="UserServlet" method="post">
    <label for="u_id">User ID:</label>
    <input type="number" id="u_id" name="u_id" required>
    <br>
    <label for="u_name">Name:</label>
    <input type="text" id="u_name" name="u_name" required>
    <br>
    <label for="u_email">Email:</label>
    <input type="email" id="u_email" name="u_email" required>
    <br>
    <label for="u_password">Password:</label>
    <input type="password" id="u_password" name="u_password" required>
    <br>

    <br>
    <input type="submit" value="Add User">
</form>
<br>
<a href="result.jsp">See All Users</a>

<%
    String errorMessage = (String) request.getAttribute("errorMessage");
    if (errorMessage != null) {
%>
<script>
    showError("<%= errorMessage %>");
</script>
<%
    }
%>
</body>
</html>
```

- Now we will work on UserServlet for for Cru Operation
- Add new Class UserServlet.java and com.example package and add the following code

```
package com.example;

import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class UserServlet extends HttpServlet {
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        String action = request.getParameter("action");

        try (Connection connection = DatabaseConnection.getConnection()) {
            if ("delete".equals(action)) {
                String u_id = request.getParameter("u_id");
                int id = Integer.parseInt(u_id);

                PreparedStatement stmt = connection.prepareStatement("DELETE
FROM users WHERE u_id = ?");
                stmt.setInt(1, id);
                stmt.executeUpdate();
                response.sendRedirect("result.jsp");
            } else if ("update".equals(action)) {
                String u_id = request.getParameter("u_id");
                String u_name = request.getParameter("u_name");
                String u_email = request.getParameter("u_email");
                String u_password = request.getParameter("u_password");
                int id = Integer.parseInt(u_id);

                PreparedStatement stmt = connection.prepareStatement(
                        "UPDATE users SET u_name = ?, u_email = ?, u_password
= ? WHERE u_id = ?");
                stmt.setString(1, u_name);
                stmt.setString(2, u_email);
                stmt.setString(3, u_password);
                stmt.setInt(4, id);
                stmt.executeUpdate();
                response.sendRedirect("result.jsp");

            } else {
                // Handle user creation
                String u_id = request.getParameter("u_id");
                String u_name = request.getParameter("u_name");
                String u_email = request.getParameter("u_email");
                String u_password = request.getParameter("u_password");

                // Check for existing ID
                PreparedStatement checkIdStmt =
connection.prepareStatement("SELECT COUNT(*) FROM users WHERE u_id = ?");
                checkIdStmt.setInt(1, Integer.parseInt(u_id));
                ResultSet rsId = checkIdStmt.executeQuery();
                rsId.next();
                boolean idExists = rsId.getInt(1) > 0;

                // Check for existing Email
                PreparedStatement checkEmailStmt =
connection.prepareStatement("SELECT COUNT(*) FROM users WHERE u_email = ?");
                checkEmailStmt.setString(1, u_email);
                ResultSet rsEmail = checkEmailStmt.executeQuery();
                rsEmail.next();
                boolean emailExists = rsEmail.getInt(1) > 0;

                if (idExists || emailExists) {
                    String errorMessage = "Please use a different ID and
email, as they are already used.";
                    request.setAttribute("errorMessage", errorMessage);

request.getRequestDispatcher("index.jsp").forward(request, response);
                } else {
                    // Insert new user
                    PreparedStatement stmt = connection.prepareStatement(
                            "INSERT INTO users (u_id, u_name, u_email,
u_password) VALUES (?, ?, ?, ?)");
                    stmt.setInt(1, Integer.parseInt(u_id));
                    stmt.setString(2, u_name);
                    stmt.setString(3, u_email);
                    stmt.setString(4, u_password);
                    stmt.executeUpdate();
                    response.sendRedirect("result.jsp");
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
            request.setAttribute("errorMessage", "Database error occurred.
Please try again.");
            request.getRequestDispatcher("index.jsp").forward(request,
response);
        } catch (NumberFormatException e) {
            e.printStackTrace();
            request.setAttribute("errorMessage", "Invalid ID format. Please
enter a valid number.");
            request.getRequestDispatcher("index.jsp").forward(request,
response);
        }
    }
}
```

- Now we will add Result.jsp and Edit.jsp codes

- Result.jsp code is follows

```
<%@ page import="java.sql.Connection" %>
<%@ page import="com.example.DatabaseConnection" %>
<%@ page import="java.sql.PreparedStatement" %>
<%@ page import="java.sql.ResultSet" %>
<%
  Connection connection = DatabaseConnection.getConnection();
  PreparedStatement stmt = connection.prepareStatement("SELECT * FROM
users");
  ResultSet rs = stmt.executeQuery();
%>

<table border="1">
  <tr>
    <th>ID</th>
    <th>Name</th>
    <th>Email</th>
    <th>Role</th>
    <th>Actions</th>
  </tr>

  <%
    while (rs.next()) {
  %>
  <tr>
    <td><%= rs.getInt("u_id") %></td>
    <td><%= rs.getString("u_name") %></td>
    <td><%= rs.getString("u_email") %></td>
    <td><%= rs.getString("u_role") %></td>
    <td>
      <form action="edit.jsp" method="get">
        <input type="hidden" name="u_id" value="<%= rs.getInt("u_id") %>">
        <input type="hidden" name="u_name" value="<%= rs.getString("u_name")
%>">
        <input type="hidden" name="u_email" value="<%=
rs.getString("u_email") %>">
        <button type="submit">Edit</button>
      </form>
      <form action="UserServlet" method="post" style="display:inline;">
        <input type="hidden" name="u_id" value="<%= rs.getInt("u_id") %>">
        <button type="submit" name="action" value="delete">Delete</button>
      </form>
    </td>
  </tr>
  <%
    }
    rs.close();
    stmt.close();
    connection.close();
  %>
</table>
```

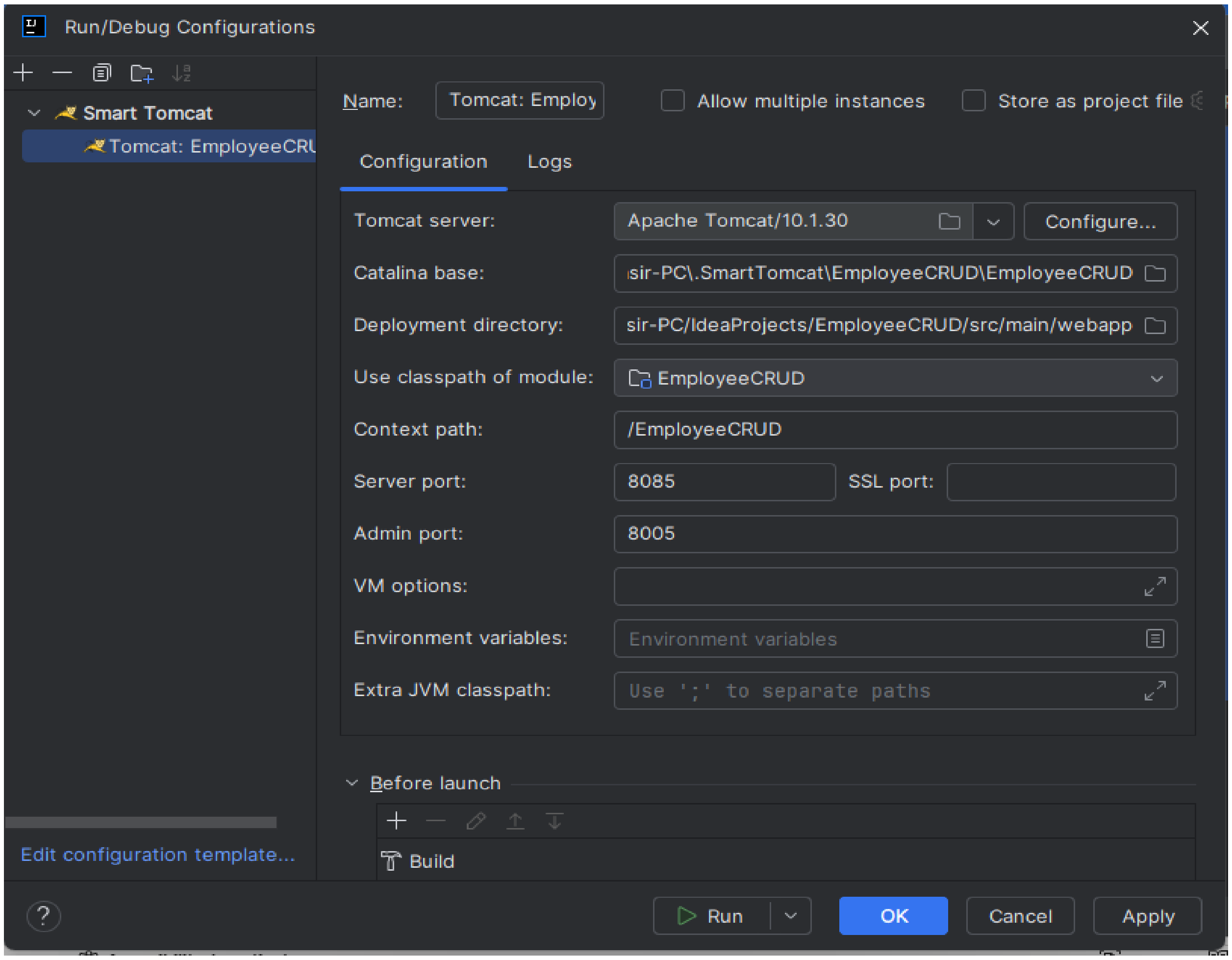
➢ Now add edit.jsp code

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%
  String u_id = request.getParameter("u_id");
  String u_name = request.getParameter("u_name");
  String u_email = request.getParameter("u_email");
%>
<h1>Edit User</h1>
<form action="UserServlet" method="post">
  <input type="hidden" name="u_id" value="<%= u_id %>">
  <label>Name:</label>
  <input type="text" name="u_name" value="<%= u_name != null ? u_name :
"" %>" required>
  <br>
  <label>Email:</label>
  <input type="email" name="u_email" value="<%= u_email != null ?
u_email : "" %>" required>
  <br>
  <label>Password:</label>
  <input type="password" name="u_password" required>
  <br><br>
  <input type="hidden" name="action" value="update">
  <input type="submit" value="Update User">
</form>
```

- Now we will Map the Servlet in Web.xml. add the following code

```
<!DOCTYPE web-app PUBLIC
        "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
        "http://java.sun.com/dtd/web-app_2_3.dtd" >

<web-app>

  <servlet>
    <servlet-name>UserServlet</servlet-name>
    <servlet-class>com.example.UserServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>UserServlet</servlet-name>
    <url-pattern>/UserServlet</url-pattern>
  </servlet-mapping>



</web-app>
```

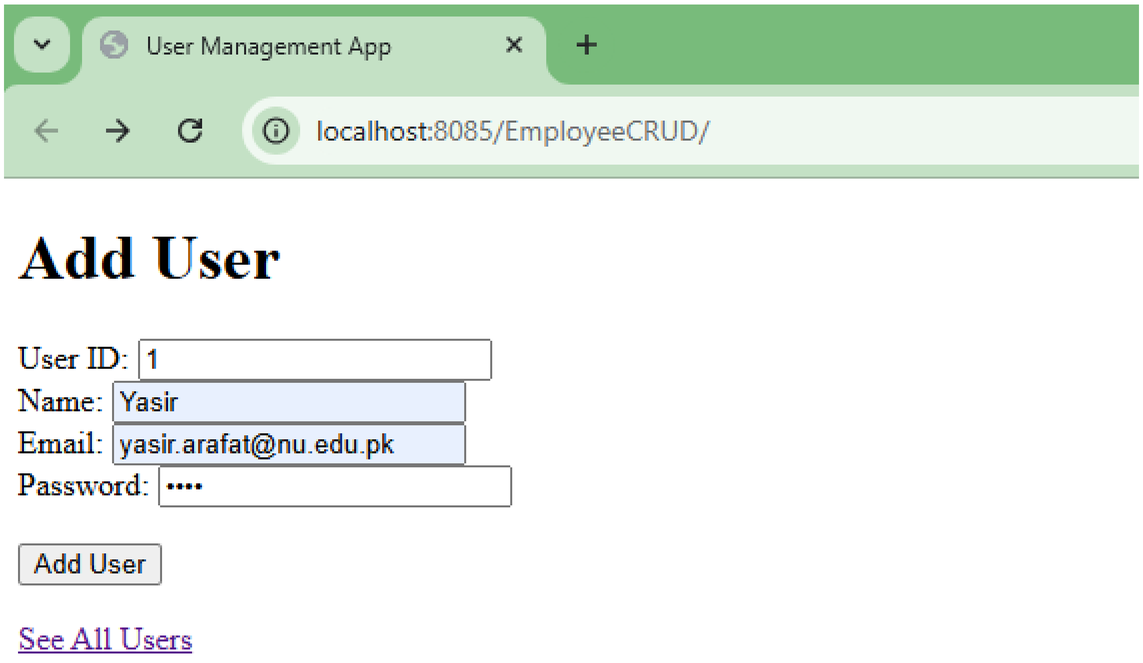
- Now run Configure your tomcat make sure to configure the userservlet not the .jsp file



Name: Tomcat: Employ
Allow multiple instances
Store as project file

Tomcat server: Apache Tomcat/10.1.30 Configure...
Catalina base: sir-PC\.SmartTomcat\EmployeeCRUD\EmployeeCRUD
Deployment directory: sir-PC/IdeaProjects/EmployeeCRUD/src/main/webapp
Use classpath of module: EmployeeCRUD
Context path: /EmployeeCRUD
Server port: 8085 SSL port:
Admin port: 8005
VM options:
Environment variables: Environment variables
Extra JVM classpath: Use ';' to separate paths

- After configuring run project paste this link on browser http://localhost:8085/EmployeeCRUD/



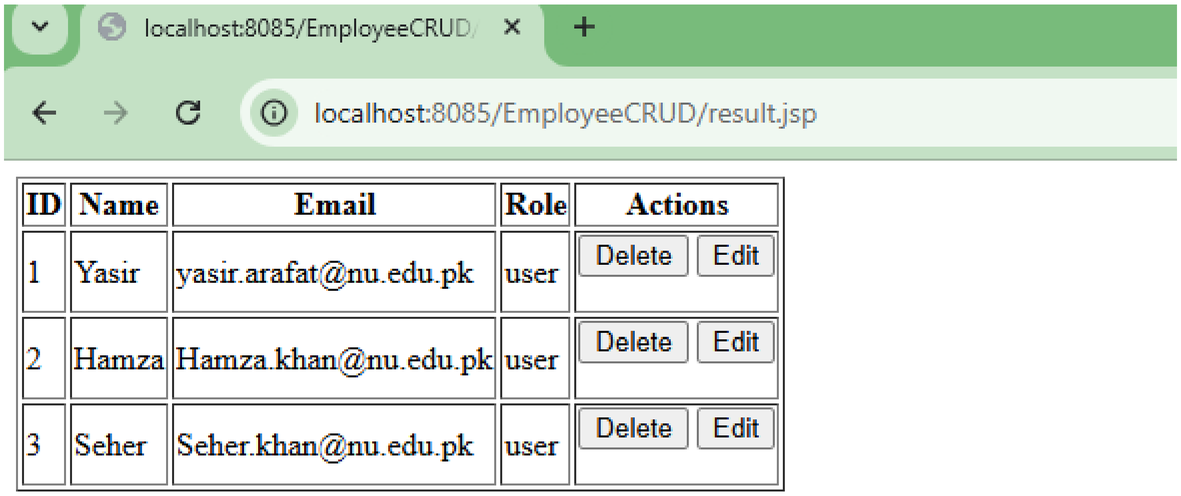
Add User

User ID: 1
Name: Yasir
Email: yasir.arafat@nu.edu.pk
Password: ••••

Add User

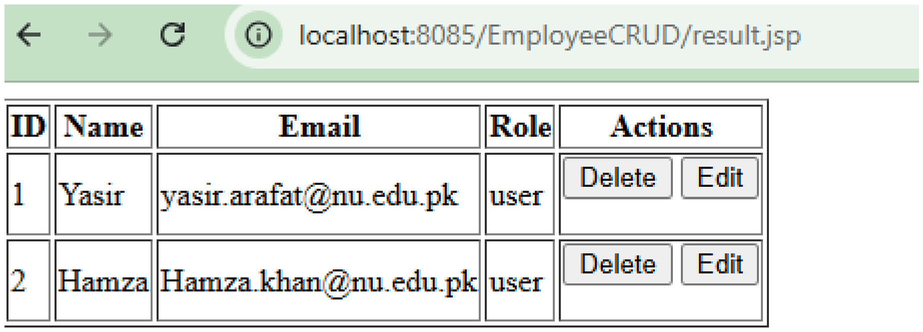
See All Users

➢ Add two or three user and then click on the see all users


localhost:8085/EmployeeCRUD/
localhost:8085/EmployeeCRUD/result.jsp

ID	Name	Email	Role	Actions
1	Yasir	yasir.arafat@nu.edu.pk	user	Delete Edit
2	Hamza	Hamza.khan@nu.edu.pk	user	Delete Edit
3	Seher	Seher.khan@nu.edu.pk	user	Delete Edit

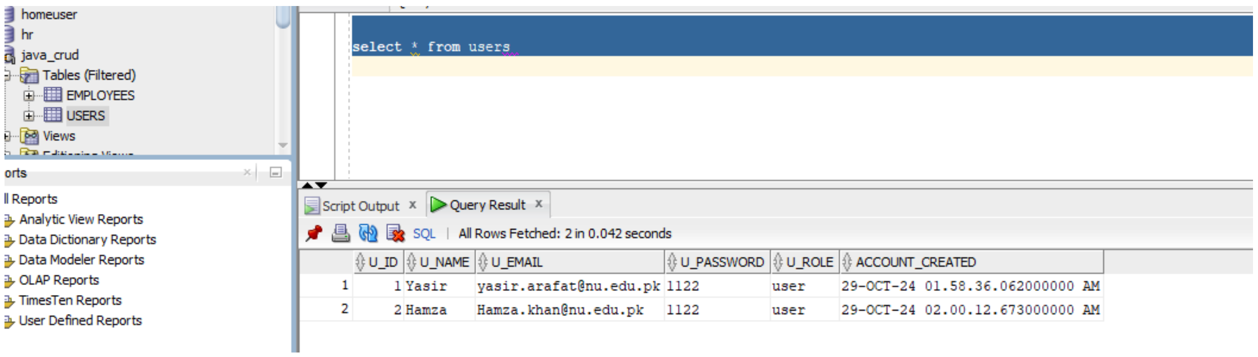
- Now you can delete user by clicking on delete



localhost:8085/EmployeeCRUD/result.jsp

ID	Name	Email	Role	Actions
1	Yasir	yasir.arafat@nu.edu.pk	user	Delete Edit
2	Hamza	Hamza.khan@nu.edu.pk	user	Delete Edit

- We can verify this from database

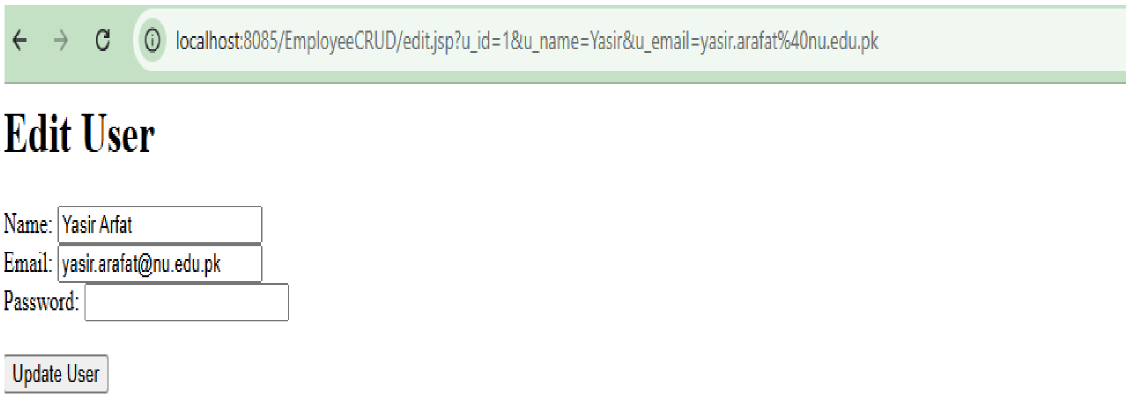


```
select * from users
```

All Rows Fetched: 2 in 0.042 seconds

```
     U_ID U_NAME  U_EMAIL                U_PASSWORD U_ROLE ACCOUNT_CREATED
1       1 Yasir   yasir.arafat@nu.edu.pk 1122       user   29-OCT-24 01.58.36.062000000 AM
2       2 Hamza   Hamza.khan@nu.edu.pk   1122       user   29-OCT-24 02.00.12.673000000 AM
```

- We can also edit our data using form

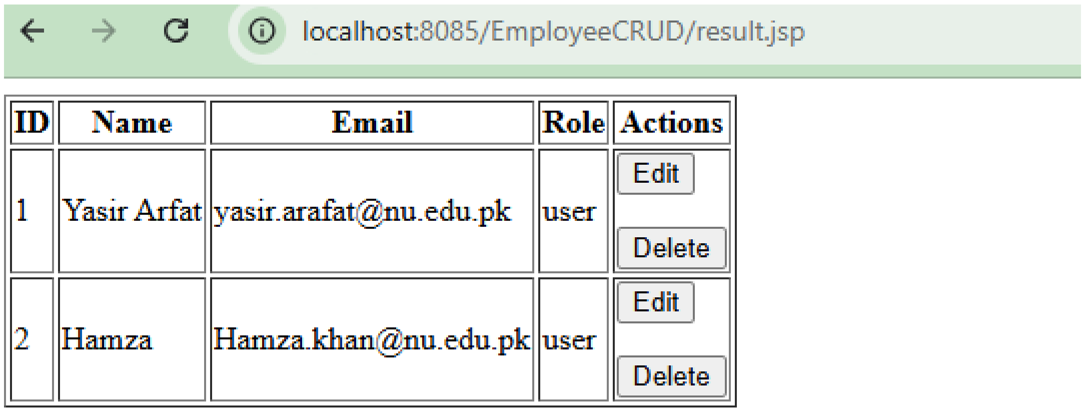


localhost:8085/EmployeeCRUD/edit.jsp?u_id=1&u_name=Yasir&u_email=yasir.arafat%40nu.edu.pk

Edit User

Name: Yasir Arfat
Email: yasir.arafat@nu.edu.pk
Password:

Update User



localhost:8085/EmployeeCRUD/result.jsp

ID	Name	Email	Role	Actions
1	Yasir Arfat	yasir.arafat@nu.edu.pk	user	Edit Delete
2	Hamza	Hamza.khan@nu.edu.pk	user	Edit Delete

Sessions and Cookies in Java

In Java web applications, **sessions** and **cookies** are used to manage user state and maintain data across multiple HTTP requests since HTTP is inherently stateless. Here's an overview of both:

Sessions in Java

A **session** represents a series of interactions between the user and the server. It allows the server to track user information (like login status) across multiple requests.

Creating a session:
In Java, sessions are managed using the HttpSession interface.

```
HttpSession session = request.getSession();  // Creates a new session or retrieves an existing one

session.setAttribute("username", "Ali");     // Store data in the session
```

Retrieving session data:

```
String username = (String) session.getAttribute("username");
```

Invalidating the session (Logout):

```
session.invalidate();  // Ends the session

session.setMaxInactiveInterval(300);  // Timeout in seconds (e.g., 5 minutes)
```

Use Case:

- User logs in, and their information is stored in the session.
- On every request, the server checks if a valid session exists to determine if the user is logged in.

2. Cookies in Java

A **cookie** is a small piece of data stored on the client-side (browser) and sent to the server with every request. Cookies help the server remember user preferences or state across visits.

- **Creating a cookie:**

```
Cookie cookie = new Cookie("theme", "dark");  // Create a cookie
cookie.setMaxAge(24 * 60 * 60);              // Expire in one day
response.addCookie(cookie);                  // Add it to the response
```

- **Retrieving cookies:**

```
Cookie[] cookies = request.getCookies();  // Get all cookies from the request
if (cookies != null) {
   for (Cookie c : cookies) {
      if (c.getName().equals("theme")) {
         System.out.println("Theme: " + c.getValue());
      }
   }
}
```

- **Deleting a cookie:**
  To delete a cookie, set its expiration time to 0:

```
Cookie cookie = new Cookie("theme", "");
cookie.setMaxAge(0);  // Delete the cookie
response.addCookie(cookie);
```

Use Case:

- A user selects a theme preference (dark mode), and it is saved in a cookie.
- On future visits, the website reads the cookie and applies the selected theme automatically.

Difference Between Sessions and Cookies

Aspect	**Session**	**Cookie**
Storage	Server-side	Client-side (in the browser)
Security	More secure (not exposed to the client)	Less secure, can be manipulated by the user

Aspect	**Session**	**Cookie**
Data Size	Can store large data	Limited to 4KB
Lifetime	Ends with session expiration or logout	Can persist even after the browser is closed
Use Case	Authentication, shopping carts, user sessions	Preferences like themes, language, etc.

Web-App with Sessions and Cookies

Here, we will use the same project to demonstrate the use of sessions and cookies. We will implement user login, dashboard, and logout functionality using sessions to manage user authentication and cookies to store preferences or track login status across visits

- Here in the above project we add the userlogin.jsp fille and code is given as

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
  <title>User Login</title>
</head>
<body>
<h2>User Login</h2>
<form action="UserLoginServlet" method="post">
  <label>Email:</label>
  <input type="email" name="email" required><br>
  <label>Password:</label>
  <input type="password" name="password" required><br>
  <input type="submit" value="Login">
</form>
</body>
</html>
```

- Now we Add the UserLoginServlet class code

```java
package com.example;

import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.http.HttpSession;
import jakarta.servlet.http.Cookie;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class UserLoginServlet extends HttpServlet {
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        String email = request.getParameter("email");
        String password = request.getParameter("password");

        try (Connection connection = DatabaseConnection.getConnection();
             PreparedStatement stmt = connection.prepareStatement(
                     "SELECT * FROM users WHERE u_email = ? AND u_password =
?")) {

            stmt.setString(1, email);
            stmt.setString(2, password);

            ResultSet rs = stmt.executeQuery();

            if (rs.next()) {
                // User is authenticated
                HttpSession session = request.getSession();
                session.setAttribute("userName", rs.getString("u_name")); //
Store username in session
                session.setAttribute("userEmail", email); // Store email in
session

                // Set a cookie with the user's email
                Cookie emailCookie = new Cookie("userEmail", email);
                emailCookie.setMaxAge(60 * 60 * 24); // 1 day
                response.addCookie(emailCookie);

                response.sendRedirect("UserDashboard.jsp");
            } else {
                // Invalid credentials
                PrintWriter out = response.getWriter();
                out.println("<script>alert('Invalid email or password!');
window.location='userlogin.jsp';</script>");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

➢ Now we will add userDashboard.jsp code

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ page session="true" %>
<%
    // Check if the user is authenticated by verifying session attributes
    String userName = (String) session.getAttribute("userName");
    String userEmail = (String) session.getAttribute("userEmail");

    if (userName == null || userEmail == null) {
        // If the session attributes are missing, redirect to the login page
        response.sendRedirect("userlogin.jsp");
        return; // Prevent further processing of the page
    }
%>
<!DOCTYPE html>
<html>
<head>
    <title>User Dashboard</title>
</head>
<body>
<h2>Welcome, <%= userName %>!</h2>
<p><%= userEmail %>!</p>
<h3>User Dashboard</h3>
<p>This is your dashboard. You can manage your profile here.</p>

<form action="LogoutServlet" >
    <input type="submit" value="Logout">
</form>
</body>
</html>
```

➢ Now we will add the logout servlet code

```
package com.example;

import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.http.HttpSession;

import java.io.IOException;
```

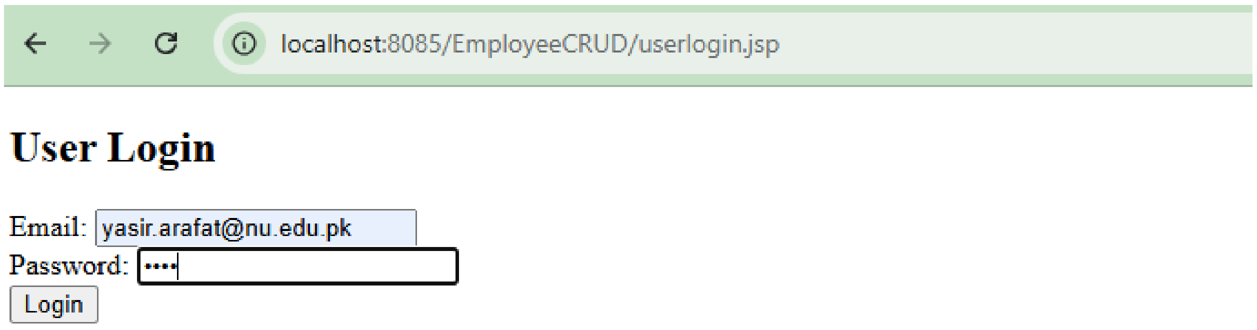
```java
public class LogoutServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse 
response) throws ServletException, IOException {
        HttpSession session = request.getSession(false);
        if (session != null) {
            session.invalidate();
        }
        response.sendRedirect("userlogin.jsp");
    }
}
```

- Now update you web.xml code

```xml
<!DOCTYPE web-app PUBLIC
        "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
        "http://java.sun.com/dtd/web-app_2_3.dtd" >

<web-app>

  <servlet>
    <servlet-name>UserServlet</servlet-name>
    <servlet-class>com.example.UserServlet</servlet-class>
  </servlet>

  <servlet>
    <servlet-name>UserLoginServlet</servlet-name>
    <servlet-class>com.example.UserLoginServlet</servlet-class>
  </servlet>

  <servlet>
    <servlet-name>LogoutServlet</servlet-name>
    <servlet-class>com.example.LogoutServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>LogoutServlet</servlet-name>
    <url-pattern>/LogoutServlet</url-pattern>
  </servlet-mapping>

  <servlet-mapping>
    <servlet-name>UserLoginServlet</servlet-name>
    <url-pattern>/UserLoginServlet</url-pattern>
  </servlet-mapping>

  <servlet-mapping>
    <servlet-name>UserServlet</servlet-name>
    <url-pattern>/UserServlet</url-pattern>
  </servlet-mapping>



</web-app>
```

- All Done Now Run Your Project

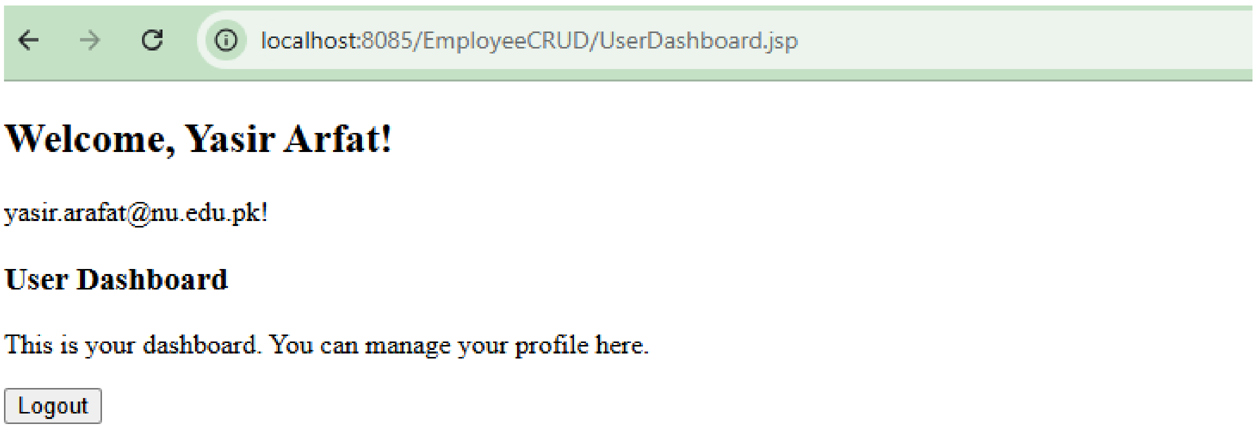
  Use tis link to check login page http://localhost:8085/EmployeeCRUD/userlogin.jsp



localhost:8085/EmployeeCRUD/userlogin.jsp

User Login

Email: yasir.arafat@nu.edu.pk
Password: ••••



localhost:8085/EmployeeCRUD/UserDashboard.jsp

Welcome, Yasir Arfat!

yasir.arafat@nu.edu.pk!

User Dashboard

This is your dashboard. You can manage your profile here.