EXPERIMENT NO. 08

Title: Create a Login and Logout System Using the Struts Validation Framework..

Outcome: Students will be able to create a Login and Logout System Using the Struts Validation Framework.

Theory:

Build a Struts 2 web app with **Login** and **Logout** functionality backed by **MySQL**, running on **Eclipse + Tomcat 9 + Java 8**.

Project name

StrutsDBAuthDemo

Prerequisites

- Eclipse (EE)
- Apache Tomcat 9.x
- Java JDK 1.8
- MySQL server (or XAMPP/WAMP)
- Struts 2.5.x JARs in WEB-INF/lib
- mysql-connector-java-8.x.x.jar in WEB-INF/lib

MySQL (Database) Setup

Run these SQL commands:

```
CREATE DATABASE strutsdb;
USE strutsdb;

CREATE TABLE users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  username VARCHAR(50) NOT NULL UNIQUE,
  password VARCHAR(255) NOT NULL
);

-- example user (plaintext password 'admin123' for demo)
INSERT INTO users (username, password) VALUES ('admin', 'admin123');
```

Security note: For production, store hashed passwords (BCrypt). I'll show an optional note on hashing at the end.

Project structure

```
StrutsDBAuthDemo/
 ├ src/
 │  └ com/example/
 │     ├ action/
 │     │  ├ LoginAction.java
 │     │  └ LogoutAction.java
 │     └ dao/
 │        └ DatabaseConnection.java
 ├ WebContent/
 │  ├ WEB-INF/
 │  │  ├ lib/
 │  │  ├ struts.xml
 │  │  └ web.xml
 │  ├ index.jsp        <-- login page
 │  ├ dashboard.jsp    <-- protected page after login
 │  ├ header.jsp       <-- common header with logout link (include)
 │  └ loginerror.jsp
```

Classes & files (copy/paste)

src/com/example/dao/DatabaseConnection.java
(unchanged from earlier; update DB creds if needed)

```
package com.example.dao;

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

public class DatabaseConnection {
   private static final String URL =
"jdbc:mysql://localhost:3306/strutsdb?useSSL=false&serverTimezone=UTC";
   private static final String USER = "root";
   private static final String PASSWORD = ""; // set your MySQL password

   public static Connection getConnection() {
      Connection con = null;
      try {
         Class.forName("com.mysql.cj.jdbc.Driver");
         con = DriverManager.getConnection(URL, USER, PASSWORD);
      } catch (Exception e) {
         e.printStackTrace();
```

```
        }
        return con;
    }
}
```

---

src/com/example/action/LoginAction.java
(implements SessionAware to add user to session)

```
package com.example.action;

import com.opensymphony.xwork2.ActionSupport;
import com.example.dao.DatabaseConnection;
import com.opensymphony.xwork2.ActionContext;
import org.apache.struts2.interceptor.SessionAware;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.Map;

public class LoginAction extends ActionSupport implements SessionAware {

    private String username;
    private String password;
    private Map<String, Object> session;

    public String getUsername() { return username; }
    public void setUsername(String username) { this.username = username; }

    public String getPassword() { return password; }
    public void setPassword(String password) { this.password = password; }

    @Override
    public String execute() {
        // default action: redirect to input (login form) if not POST
        return INPUT;
    }

    public String validateLogin() {
        try (Connection con = DatabaseConnection.getConnection()) {
            PreparedStatement ps = con.prepareStatement(
                "SELECT * FROM users WHERE username=? AND password=?");
```

```
            ps.setString(1, username);
            ps.setString(2, password);
            ResultSet rs = ps.executeQuery();

            if (rs.next()) {
                // login success -> put username in session
                session.put("USER", username);
                return SUCCESS;
            } else {
                addActionError("Invalid username or password.");
                return INPUT;
            }
        } catch (Exception e) {
            e.printStackTrace();
            addActionError("Server error. Try again later.");
            return ERROR;
        }
    }

    // SessionAware implementation
    @Override
    public void setSession(Map<String, Object> session) {
        this.session = session;
    }
}
```

Notes:

- I separated execute() (default) and validateLogin() (the login POST action). Alternatively you can use execute() for login logic; validateLogin clarifies intent.

---

src/com/example/action/LogoutAction.java

```
package com.example.action;

import com.opensymphony.xwork2.ActionSupport;
import org.apache.struts2.interceptor.SessionAware;

import java.util.Map;

public class LogoutAction extends ActionSupport implements SessionAware {

    private Map<String, Object> session;
```

```
    @Override
    public String execute() {
        if (session != null) {
            session.remove("USER");
            session.clear(); // remove all session attributes
        }
        return SUCCESS; // will redirect to index.jsp or login page
    }

    @Override
    public void setSession(Map<String, Object> session) {
        this.session = session;
    }
}
```

WebContent/WEB-INF/struts.xml

Add both login and logout actions:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.5//EN"
    "http://struts.apache.org/dtds/struts-2.5.dtd">

<struts>
    <package name="default" namespace="/" extends="struts-default">

        <!-- show login form -->
        <action name="index" class="com.example.action.LoginAction"
method="execute">
            <result name="input">/index.jsp</result>
        </action>

        <!-- handle login POST -->
        <action name="login" class="com.example.action.LoginAction"
method="validateLogin">
            <result name="success">/dashboard.jsp</result>
            <result name="input">/index.jsp</result>
            <result name="error">/loginerror.jsp</result>
        </action>

        <!-- logout -->
```

```
    <action name="logout" class="com.example.action.LogoutAction"
method="execute">
      <result name="success">/index.jsp</result>
    </action>

  </package>
</struts>
```

WebContent/WEB-INF/web.xml

(ensure correct filter class — no typos)

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
                  http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
    version="3.1">

  <display-name>Struts DB Auth Demo</display-name>

  <filter>
    <filter-name>struts2</filter-name>
    <filter-
class>org.apache.struts2.dispatcher.filter.StrutsPrepareAndExecuteFilter</filter-class>
  </filter>

  <filter-mapping>
    <filter-name>struts2</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>

  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>

</web-app>
```

JSP Pages

WebContent/index.jsp (login page)

```
<%@ taglib prefix="s" uri="/struts-tags" %>
<%@ include file="header.jsp" %>
<html>
```

```
<head><title>Login</title></head>
<body>
   <h2>Login</h2>

   <s:form action="login" method="post">
      <s:textfield name="username" label="Username" />
      <s:password name="password" label="Password" />
      <s:submit value="Login" />
   </s:form>

   <s:actionerror />
   <s:fielderror fieldName="username" />
   <s:fielderror fieldName="password" />
</body>
</html>
```

WebContent/header.jsp (common header — shows logout when logged in)

```
<%@ taglib prefix="s" uri="/struts-tags" %>
<%@ page session="true" %>
<div style="background:#eee;padding:8px;margin-bottom:10px;">
   <a href="index">Home</a> |
   <s:if test="%{#session.USER != null}">
      Welcome, <s:property value="#session.USER" /> |
      <a href="logout">Logout</a>
   </s:if>
</div>
```

WebContent/dashboard.jsp (protected page)

```
<%@ taglib prefix="s" uri="/struts-tags" %>
<%@ include file="header.jsp" %>
<html>
<head><title>Dashboard</title></head>
<body>
   <!-- basic session check -->
   <s:if test="%{#session.USER == null}">
      <s:action name="index" />
   </s:if>

   <h2>Dashboard</h2>
   <p>You are logged in as: <s:property value="#session.USER" /></p>
</body>
```

```
</html>
```

Note: The <s:if> above performs a simple check and redirects to index action if the user is not logged in. For stricter protection, use a Servlet Filter or Struts interceptor.

WebContent/loginerror.jsp

```
<html>
<head><title>Error</title></head>
<body>
   <h2>Server error occurred. Try again later.</h2>
   <a href="index">Back to Login</a>
</body>
</html>
```

Flow summary

- index action → shows index.jsp (login form)
- login action → reads credentials, checks DB; on success writes USER to session and sends to dashboard.jsp
- dashboard.jsp checks session and displays username & logout link
- logout action clears session and redirects to index.jsp

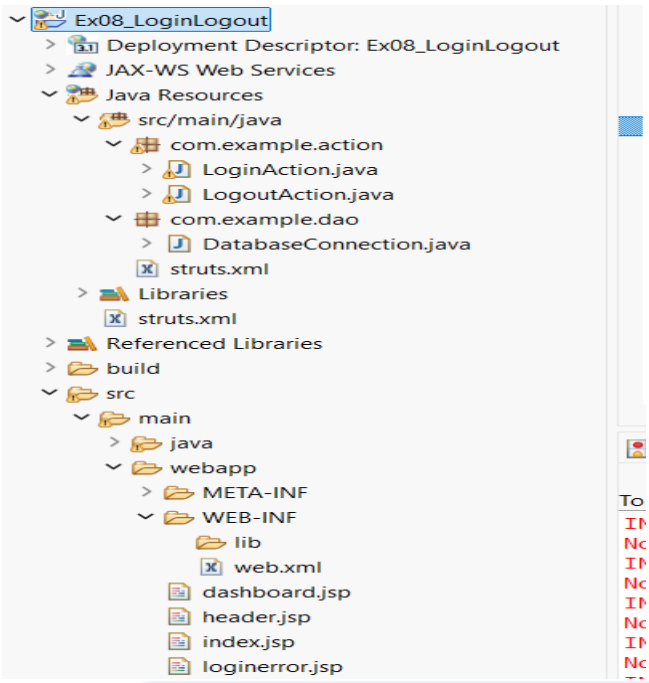
Deployment steps (Eclipse)

1. Create **Dynamic Web Project** StrutsDBAuthDemo.
2. Add Struts + MySQL connector JARs to WebContent/WEB-INF/lib.
3. Create Java packages & classes (dao, action) and paste code above.
4. Add struts.xml and web.xml under WebContent/WEB-INF/.
5. Create JSPs in WebContent/.
6. Project → Clean.
7. Right-click → Run As → **Run on Server** (Tomcat 9).
8. Open: http://localhost:8080/StrutsDBAuthDemo/ (or path shown by Eclipse).

Security recommendations (quick)

1. **Hash passwords**: don't store plaintext. Use BCrypt (recommended). See *Optional* below for snippet.
2. Use prepared statements (we already do).
3. Use HTTPS (TLS) in production.
4. Limit session timeouts — configure session-config in web.xml if desired.
5. Use a connection pool (Apache DBCP/HikariCP) for production instead of DriverManager.

Output:-



Conclusion : Thus, I have created a Login and Logout System Using the Struts Validation Framework.