EXPERIMENT NO. 07

Title: Build a Web Application Using the Struts Framework with Database Integration.

Outcome: Students will be able to develop a Web Application Using the Struts Framework with Database Integration.

Theory:

Prerequisites

- Eclipse (EE)
- Apache Tomcat 9.x
- Java JDK 1.8
- MySQL server (or XAMPP)
- Struts 2.5.x JARs (put in WebContent/WEB-INF/lib)
- mysql-connector-java-8.x.x.jar (in WEB-INF/lib)
- Optional (for hashing): jBCrypt-0.4.jar (or org.mindrot:jbcrypt)

---

1) Database (MySQL)

Run these commands:

```
CREATE DATABASE strutsdb;
USE strutsdb;

CREATE TABLE users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  username VARCHAR(50) NOT NULL UNIQUE,
  email VARCHAR(100) NOT NULL UNIQUE,
  password VARCHAR(255) NOT NULL,
  dob DATE,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- (optional) Example: admin user with plaintext (demo only)
INSERT INTO users (username, email, password, dob) VALUES
('admin','admin@example.com','admin123','1990-01-01');
```

Important: In production, store hashed passwords (see BCrypt section below).

---

2) Project structure (Eclipse Dynamic Web Project)

```
StrutsRegistrationDemo/
 ├ src/
 │  └ com/example/
 │     ├ action/
```

```
│   │   ├ RegistrationAction.java
│   │   └ LoginAction.java    (optional, for login)
│   └ dao/
│       └ DatabaseConnection.java
├ WebContent/
│   ├ WEB-INF/
│   │   ├ lib/      (all jars)
│   │   ├ struts.xml
│   │   └ web.xml
│   ├ index.jsp     (links to register/login)
│   ├ register.jsp
│   ├ register-success.jsp
│   ├ login.jsp     (optional)
│   └ login-success.jsp (optional)
```

---

3) Add required JARs (copy to WebContent/WEB-INF/lib)

- struts2-core-2.5.x.jar and dependencies: xwork-core, ognl, freemarker, commons-io, commons-lang3, etc.
- mysql-connector-java-8.x.x.jar
- Optional: jBCrypt-0.4.jar if you use BCrypt

---

4) Database helper

File: src/com/example/dao/DatabaseConnection.java

```
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 DB 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;
    }
}
```

---

5) Registration Action (with server-side validation)
File: src/com/example/action/RegistrationAction.java

```
package com.example.action;

import com.opensymphony.xwork2.ActionSupport;
import com.example.dao.DatabaseConnection;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Date;
import java.text.SimpleDateFormat;

public class RegistrationAction extends ActionSupport {
    private String username;
    private String email;
    private String password;
    private String dob; // dd-MM-yyyy from form

    // getters & setters
    public String getUsername() { return username; }
    public void setUsername(String username) { this.username = username; }
    public String getEmail() { return email; }
    public void setEmail(String email) { this.email = email; }
    public String getPassword() { return password; }
    public void setPassword(String password) { this.password = password; }
    public String getDob() { return dob; }
    public void setDob(String dob) { this.dob = dob; }

    @Override
    public String execute() {
        // show form (if direct access)
        return INPUT;
    }
```

```
public String register() {
    Connection con = null;
    try {
        con = DatabaseConnection.getConnection();

        // check username/email uniqueness
        PreparedStatement check = con.prepareStatement(
            "SELECT id FROM users WHERE username=? OR email=?");
        check.setString(1, username);
        check.setString(2, email);
        ResultSet rs = check.executeQuery();
        if (rs.next()) {
            addActionError("Username or Email already exists.");
            return INPUT;
        }

        // Convert dob (string) to java.sql.Date
        Date sqlDob = null;
        if (dob != null && !dob.trim().isEmpty()) {
            SimpleDateFormat sdf = new SimpleDateFormat("dd-MM-yyyy");
            java.util.Date parsed = sdf.parse(dob);
            sqlDob = new Date(parsed.getTime());
        }

        // If using plaintext (demo) — **replace with hashed password in production**
        PreparedStatement ps = con.prepareStatement(
            "INSERT INTO users (username, email, password, dob) VALUES (?,?,?,?)");
        ps.setString(1, username);
        ps.setString(2, email);
        ps.setString(3, password); // TODO: replace with hashed value
        ps.setDate(4, sqlDob);
        int inserted = ps.executeUpdate();

        if (inserted > 0) {
            return SUCCESS;
        } else {
            addActionError("Registration failed, try again.");
            return ERROR;
        }

    } catch (Exception e) {
```

```
            e.printStackTrace();
            addActionError("Server error. Contact admin.");
            return ERROR;
        } finally {
            try { if (con != null) con.close(); } catch (Exception ex) {}
        }
    }

    @Override
    public void validate() {
        // Basic server-side validation
        if (username == null || username.trim().isEmpty())
addFieldError("username","Username required");
        if (email == null || email.trim().isEmpty()) addFieldError("email","Email required");
        else if (!email.matches("^[A-Za-z0-9+_.-]+@(.+)$"))
addFieldError("email","Invalid email");
        if (password == null || password.trim().length() < 6)
addFieldError("password","Password must be >=6 chars");
        if (dob == null || dob.trim().isEmpty()) addFieldError("dob","Date of birth required
(dd-MM-yyyy)");
        else {
            try {
                SimpleDateFormat sdf = new SimpleDateFormat("dd-MM-yyyy");
                sdf.setLenient(false);
                sdf.parse(dob);
            } catch (Exception ex) {
                addFieldError("dob","Invalid date format (dd-MM-yyyy)");
            }
        }
    }
}
```

Notes:

- register() is the method mapped for POST. execute() returns INPUT to display form if accessed directly.
- For real apps, hash the password. See BCrypt section.

---

6) Login Action (optional, for flow)

File: src/com/example/action/LoginAction.java (simple)

```
package com.example.action;

import com.opensymphony.xwork2.ActionSupport;
```

```
import com.example.dao.DatabaseConnection;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.Map;
import org.apache.struts2.interceptor.SessionAware;

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; }

   public String execute() { return INPUT; }

   public String doLogin() {
      try (Connection con = DatabaseConnection.getConnection()) {
         PreparedStatement ps = con.prepareStatement(
            "SELECT id FROM users WHERE username=? AND password=?");
         ps.setString(1, username);
         ps.setString(2, password); // use hashed compare if stored hashed
         ResultSet rs = ps.executeQuery();
         if (rs.next()) {
            session.put("USER", username);
            return SUCCESS;
         } else {
            addActionError("Invalid username/password.");
            return INPUT;
         }
      } catch (Exception e) {
         e.printStackTrace();
         addActionError("Server error.");
         return ERROR;
      }
   }

   @Override
```

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

---

7) struts.xml

File: WebContent/WEB-INF/struts.xml

```
<?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">

   <!-- Registration form display -->
   <action name="registerForm" class="com.example.action.RegistrationAction"
method="execute">
     <result name="input">/register.jsp</result>
   </action>

   <!-- Registration submit -->
   <action name="register" class="com.example.action.RegistrationAction"
method="register">
     <result name="success">/register-success.jsp</result>
     <result name="input">/register.jsp</result>
     <result name="error">/register.jsp</result>
   </action>

   <!-- Login form (optional) -->
   <action name="loginForm" class="com.example.action.LoginAction"
method="execute">
     <result name="input">/login.jsp</result>
   </action>

   <!-- Login submit -->
   <action name="login" class="com.example.action.LoginAction" method="doLogin">
     <result name="success">/login-success.jsp</result>
     <result name="input">/login.jsp</result>
     <result name="error">/login.jsp</result>
   </action>

  </package>
```

```
</struts>
```

---

8) web.xml

File: WebContent/WEB-INF/web.xml

```
<?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 Registration 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>
```

---

9) JSP pages

WebContent/register.jsp

```
<%@ taglib prefix="s" uri="/struts-tags" %>
<html><head><title>Register</title></head>
<body>
<h2>User Registration</h2>

<s:form action="register" method="post">
    <s:textfield name="username" label="Username" />
    <s:textfield name="email" label="Email" />
    <s:password name="password" label="Password" />
```

```
    <s:textfield name="dob" label="DOB (dd-MM-yyyy)" />
    <s:submit value="Register" />
</s:form>

<s:actionerror />
<s:fielderror fieldName="username" />
<s:fielderror fieldName="email" />
<s:fielderror fieldName="password" />
<s:fielderror fieldName="dob" />

</body></html>
```

WebContent/register-success.jsp

```
<html><head><title>Registered</title></head>
<body>
<h2>Registration Successful ✅</h2>
<p>You may <a href="loginForm">login</a> now.</p>
</body></html>
```

WebContent/index.jsp

```
<html><head><title>Home</title></head>
<body>
<h2>Welcome</h2>
<p><a href="registerForm">Register</a> | <a href="loginForm">Login</a></p>
</body></html>
```

Optional login pages (if using LoginAction)

login.jsp, login-success.jsp similar to register.

---

10) Optional: XML-based validation

Instead of validate() in the action, you can create RegistrationAction-validation.xml under src (root of classpath):

Example RegistrationAction-validation.xml:

```
<!DOCTYPE validators PUBLIC "-//OpenSymphony Group//XWork Validator 1.0.2//EN"
    "http://www.opensymphony.com/xwork/xwork-validator-1.0.2.dtd">
<validators>
    <field name="username">
        <field-validator type="requiredstring">
            <message>Username is required</message>
        </field-validator>
    </field>
    <field name="email">
        <field-validator type="requiredstring">
```

```
        <message>Email required</message>
      </field-validator>
      <field-validator type="email">
        <message>Invalid Email</message>
      </field-validator>
    </field>
    <field name="password">
      <field-validator type="stringlength">
        <param name="minLength">6</param>
        <message>Password must be at least 6 characters</message>
      </field-validator>
    </field>
    <field name="dob">
      <field-validator type="requiredstring">
        <message>DOB is required</message>
      </field-validator>
    </field>
</validators>
```

Place this file in src (so it becomes available in classpath as RegistrationAction-validation.xml).

---

11) Password hashing with BCrypt (recommended)

Add jBCrypt-0.4.jar to WEB-INF/lib.

Modify registration insertion to hash password:

```
import org.mindrot.jbcrypt.BCrypt;
// ...
String hashed = BCrypt.hashpw(password, BCrypt.gensalt());
ps.setString(3, hashed);
```

Modify login check:

```
// get stored hash
PreparedStatement ps = con.prepareStatement("SELECT password FROM users WHERE username=?");
ps.setString(1, username);
ResultSet rs = ps.executeQuery();
if (rs.next()) {
   String storedHash = rs.getString("password");
   if (BCrypt.checkpw(password, storedHash)) {
      // success
   } else {
      // fail
   }
```

}

---

12) Deployment steps (Eclipse)

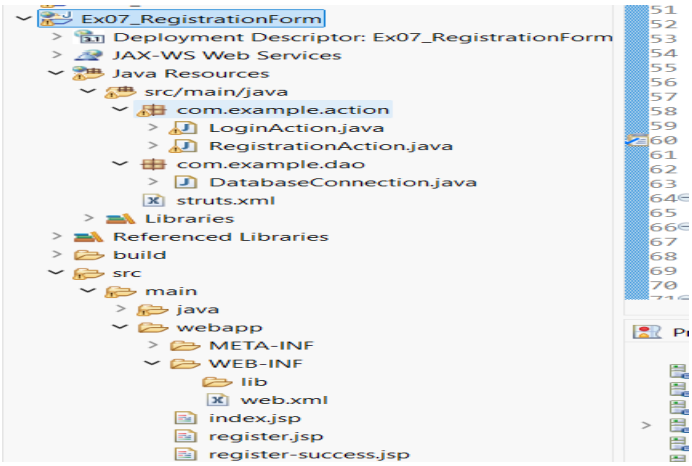
1. Create Dynamic Web Project StrutsRegistrationDemo.
2. Add Struts + JDBC + BCrypt jars to WebContent/WEB-INF/lib.
3. Create packages (com.example.dao, com.example.action) and add Java classes.
4. Add struts.xml and web.xml under WebContent/WEB-INF/.
5. Create JSPs under WebContent/.
6. Project → Clean.
7. Run on Server (Tomcat 9).
8. Open http://localhost:8080/StrutsRegistrationDemo/ → click **Register**.

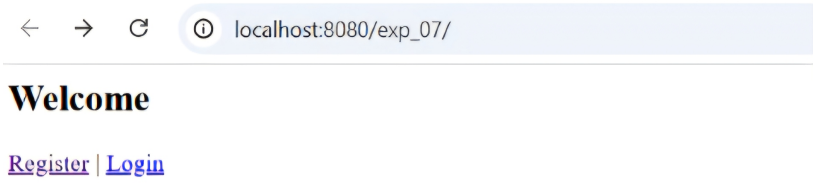
---

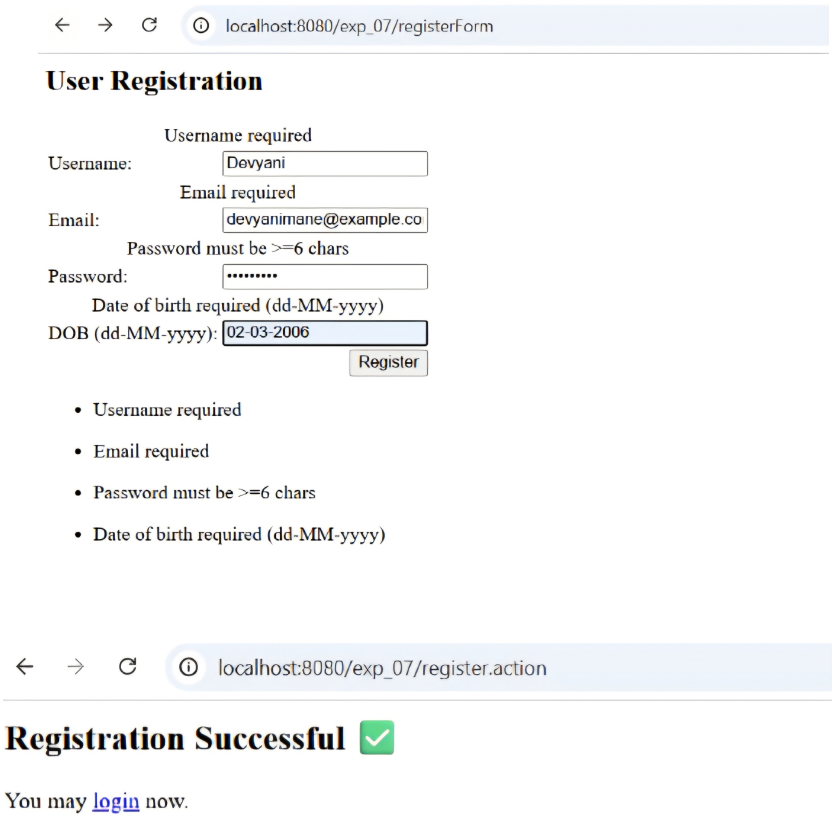
13) Testing

- Try registering with valid data. Check users table in MySQL to confirm insert.
- Try duplicate username/email — should show error.
- Try invalid email or DOB — validation should block.

Output:-







Conclusion : Thus, I have developed a Web Application Using the Struts Framework with Database Integration.