UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

NAAC GRADE A+ ACCREDITED UNIVERSITY

Worksheet (Experiment 3)

Student Name: HARSH **UID:** 24MCA20045

Branch: MCA GENERAL **section/Group-** 24MCA 1 "A"

Semester: 2nd **Date of Performance:** 11/02/2025

Subject Name: Advanced Internet Programming Lab **Subject Code:** 24CAP-652

AIM

A] Create Java EE application that maintain the employee data using ORM [hibernate] and develop the Java Persistent Query to persist the data in database.

B] use JNDI naming for Java database connectivity.

OBJECTIVE

The objective of this experiment is to develop a Java EE application that efficiently manages employee data using Object-Relational Mapping (ORM) with Hibernate. The key goals include:

- Develop a Java EE application to manage employee data using Hibernate ORM.
- Implement Java Persistence API (JPA) for defining entities and CRUD operations.
- Establish database connectivity using Hibernate configuration. • Establish a database connection and persist employee records.
- Ensure efficient transaction handling and exception management.

Task To Be Done

1. Install and configure Java EE, Hibernate, and MySQL.
2. Add required JAR dependencies (Hibernate, JPA, JDBC, MySQL).
3. Configure hibernate.cfg.xml for database connection.
4. Create an employee_db database.
5. Create Java classes (Employee.java, EmployeeDAO.java, HibernateUtil.java, MainApp.java).
6. Deploy the Java EE application on Apache server.

Implementation

Please find the below project structure, steps and implemented code in Eclipse with Apache Tomcat server configuration for above experiment.

Steps

Step 1: Create a New Maven Project

1. **Open NetBeans** and go to the **File** menu.
2. Select **New Project**.
    - In the dialog box, choose **Java** and then **Maven**.
    - Choose **Web Application** as the project type and click **Next**.
3. **Set the Project Name** and **Project Location**.
    - Project Name: EmployeeManagementApp
    - Project Location: Choose your desired directory for the project.
    - Group ID: com. employee
    - Artifact ID: EmployeeManagementApp
    - Version: 1.0-SNAPSHOT
4. Click **Finish**.

This creates the basic structure of your Maven web application project in NetBeans.

Step 2: Add Maven Dependencies for Hibernate and MySQL

1. In the **Projects** tab, navigate to the newly created EmployeeManagementApp project.
2. Right-click on the pom.xml file (located in the root directory) and select **Open**.
3. Add the necessary dependencies for **Hibernate**, **JPA**, **MySQL**, **Servlets**, and **JSTL** in the <dependencies> section of pom.xml.

```
<dependencies>
  <!-- Hibernate Core -->
```

```
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.5.6.Final</version>
</dependency>

<!-- Hibernate JPA API -->
<dependency>
    <groupId>javax.persistence</groupId>
    <artifactId>javax.persistence-api</artifactId>
    <version>2.2</version>
</dependency>
<!-- MySQL Connector -->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.23</version>
</dependency>
<!-- Servlet API -->
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>4.0.1</version>
    <scope>provided</scope>
</dependency>
<!-- JSTL -->
```

```
    <dependency>
        <groupId>javax.servlet.jsp.jstl</groupId>
        <artifactId>jstl</artifactId>
        <version>1.2</version>
    </dependency>
    <!-- C3P0 for connection pooling -->
    <dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-c3p0</artifactId>
        <version>5.5.6.Final</version>
    </dependency>
</dependencies>
```

4. **Save the pom.xml file**. NetBeans will automatically download the necessary dependencies.

Step 3: Create Hibernate Configuration File (hibernate.cfg.xml)

1. In the **Projects** tab, right-click on the **src/main/resources** folder.
2. Select **New > Other** and choose **Other > Empty File**.
3. Name the file hibernate.cfg.xml and click **Finish**.
4. Add the following Hibernate configuration to the hibernate.cfg.xml file:

```
<?xml version="1.0" encoding="utf-8"?>
<hibernate-configuration>
    <session-factory>
        <!-- JDBC Database connection settings -->
        <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
        <property name="hibernate.connection.driver_class">com.mysql.cj.jdbc.Driver</property>
```

```
        <property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/employeeDB</property>
        <property name="hibernate.connection.username">root</property>
        <property name="hibernate.connection.password">password</property>
        <!-- JDBC connection pool settings -->
        <property name="hibernate.c3p0.min_size">5</property>
        <property name="hibernate.c3p0.max_size">20</property>
        <property name="hibernate.c3p0.timeout">300</property>
        <!-- Enable Hibernate's automatic session context management -->
        <property name="hibernate.current_session_context_class">thread</property>
        <!-- Echo all executed SQL to stdout -->
        <property name="hibernate.show_sql">true</property>
        <!-- Drop and re-create the database schema on startup -->
        <property name="hibernate.hbm2ddl.auto">update</property>
        <!-- Mention annotated class -->
        <mapping class="com.employee.model.Employee"/>
    </session-factory>
</hibernate-configuration>
```

This file will configure the Hibernate session factory to connect to the MySQL database.

Step 4: Create Entity Class (Employee.java)

1. In the **Projects** tab, navigate to the src/main/java/com/employee/model package.
2. Right-click on the **model** package and choose **New > Java Class**.
3. Name the class Employee and click **Finish**.
4. Implement the Employee entity class:

```
package com.employee.model;
import javax.persistence.Entity;
```

```
import javax.persistence.Id;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Table;
@Entity
@Table(name = "employees")
public class Employee {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    private String name;
    private String department;
    private double salary;
    // Constructors, getters, and setters
    public Employee() {}
    public Employee(String name, String department, double salary) {
        this.name = name;
        this.department = department;
        this.salary = salary;
    }
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
    public String getDepartment() { return department; }
```

```
    public void setDepartment(String department) { this.department = department; }
    public double getSalary() { return salary; }
    public void setSalary(double salary) { this.salary = salary; }
}
```

Step 5: Create Service Class (EmployeeService.java)

1. In the **Projects** tab, right-click on the **service** package under src/main/java/com/employee/.
2. Choose **New > Java Class**, name it EmployeeService, and click **Finish**.
3. Implement the EmployeeService class:

```
package com.employee.service;
import com.employee.model.Employee;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;
import java.util.List;
public class EmployeeService {
    private static SessionFactory factory;
    static {
        factory = new
Configuration().configure("hibernate.cfg.xml").addAnnotatedClass(Employee.class).buildSessio
nFactory();
    }
    public void saveEmployee(Employee employee) {
        Session session = factory.getCurrentSession();
        Transaction transaction = session.beginTransaction();
```

```
        session.save(employee);
        transaction.commit();
    }

    public Employee getEmployee(int employeeId) {
        Session session = factory.getCurrentSession();
        Transaction transaction = session.beginTransaction();
        Employee employee = session.get(Employee.class, employeeId);
        transaction.commit();
        return employee;
    }
    public void deleteEmployee(int employeeId) {
        Session session = factory.getCurrentSession();
        Transaction transaction = session.beginTransaction();
        Employee employee = session.get(Employee.class, employeeId);
        session.delete(employee);
        transaction.commit();
    }
}
```

Step 6: Create Servlet (EmployeeServlet.java)

1. In the **Projects** tab, right-click on the **servlet** package under src/main/java/com/employee/.
2. Choose **New > Java Class**, name it EmployeeServlet, and click **Finish**.
3. Implement the EmployeeServlet class:

```
package com.employee.servlet;
```

```
import com.employee.model.Employee;
import com.employee.service.EmployeeService;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;
import java.io.IOException;
import java.util.List;
@WebServlet("/EmployeeServlet")
public class EmployeeServlet extends HttpServlet {
    private EmployeeService employeeService = new EmployeeService();
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        List<Employee> employees = employeeService.getAllEmployees();
        request.setAttribute("employees", employees);
        request.getRequestDispatcher("/listEmployees.jsp").forward(request, response);
    }
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        String name = request.getParameter("name");
        String department = request.getParameter("department");
        double salary = Double.parseDouble(request.getParameter("salary"));
        Employee employee = new Employee(name, department, salary);
        employeeService.saveEmployee(employee);
        response.sendRedirect("EmployeeServlet");
```

```
    }
}
```

Step 7: Create JSP Page (listEmployees.jsp)

1. In the **Projects** tab, right-click on the **webapp/WEB-INF** folder and choose **New > JSP File**.
2. Name the file listEmployees.jsp and click **Finish**.
3. Add the following code to listEmployees.jsp to display the list of employees:

```
<!DOCTYPE html>
<html>
<head>
    <title>Employee List</title>
</head>
<body>
    <h2>Employee List</h2>
    <table border="1">
        <thead>
        <tr>
            <th>ID</th>
            <th>Name</th>
            <th>Department</th>
            <th>Salary</th>
        </tr>
        </thead>
        <tbody>
        <c:forEach var="employee" items="${employees}">
```

```
            <tr>
                <td>${employee.id}</td>
                <td>${employee.name}</td>
                <td>${employee.department}</td>
                <td>${employee.salary}</td>
            </tr>
        </c:forEach>
        </tbody>
    </table>
</body>
</html>
```

Step 8: Run the Application

1. **Configure Database**: Ensure you have a **MySQL** database (e.g., employeeDB) with the necessary employees table.
2. **Build and Run**: Right-click on the project and select **Run** or **Build and Deploy**.

Once you open your browser, go to http://localhost:8080/EmployeeManagementApp/EmployeeServlet to view the employee data. You can also add new employees via the servlet form and see the updates on the list page.

Learning Outcomes (What I have Learned)

- Learned about the concept of Hibernate ORM and its role in Java EE applications.
- Learned to add JAR dependencies (Hibernate, JPA, JDBC, MySQL).
- Learned to configure Hibernate and connect it with a MySQL database.
- Learned about session and transaction management in Hibernate.
- Learned about DAO (Data Access Object) pattern for database operations.

Sr. No.	Parameters	Marks Obtained	Maximum Marks
1.	Worksheet		8 Marks
2.	Viva		10 Marks
3.	Simulation		12 Marks
	Total		30 Marks

Teacher Signature