

Adv.JAVA

Practical Lab_03

Maven hibernate project

Submitted By :-

Name:- Praveer Kishore

Enrollment No.:- A45349521001

Course:- BCA+MCA (Dual)

Semester:- 6th 'B'

About Our Project:-

Employee Management System using Hibernate

The Employee Management System is a Java application built using Maven, Hibernate, and JDBC, developed in Eclipse IDE. The system provides functionalities to manage employee records such as creating, fetching, editing, and deleting employee data from a database table. It offers a simple command-line interface for user interaction.

Key Features:

1. Create Employee:

Users can add new employee records by providing details such as name, salary, phone number, and address.

2. Fetch Employee:

Users can retrieve employee details by entering the employee ID..

3. Edit or Modify Employee Data:

Users can update existing employee information by selecting the employee ID and providing the new details for name, salary, phone, and address.

This operation is useful for updating information when a customer's phone number is transferred.

4. Delete Employee:

Users can remove employee records from the system by specifying the employee ID.

5. View Full Employee Table:

Users can view the entire list of employees stored in the database table, displaying all available attributes for each employee.

6. Interactive Command-Line Interface:

The system offers an intuitive command-line interface for user interaction, displaying menu options and prompting users for input choices.

7. Hibernate Integration:

The project utilizes Hibernate for database interaction, providing a robust and efficient ORM (Object-Relational Mapping) solution for managing Java objects and relational database tables.

8. Maven Dependency Management:

Maven is employed for project management and dependency resolution, ensuring smooth build processes and easy integration of libraries like Hibernate.

9. Database Connectivity with JDBC:

JDBC (Java Database Connectivity) is used for establishing connections with the underlying database, enabling CRUD (Create, Read, Update, Delete) operations on the employee table.

Tech Stack Used in the Telecom Management System:-

1. Integrated Development Environment (IDE):

Eclipse: The project is developed using the Eclipse IDE, providing a robust and user-friendly development environment for Java applications.

2. Database Management System:

MySQL Workbench: MySQL Workbench is used as the database management system to design, model, create, and manage the MySQL database that stores customer information.

3. Programming Language:

Java: The entire application is written in Java, a versatile and platform-independent programming language. Java is used for its object-oriented principles, extensive libraries, and wide industry adoption.

4. Java Database Connectivity (JDBC) Architecture:

JDBC is employed for database connectivity, allowing Java applications to interact with relational databases like MySQL. The JDBC architecture is utilized to establish connections, execute SQL queries, and manage data retrieval and updates.

PoJo of Telecom Management System:-

Employee

- Employee ID: Int
- Employee Name: String
- Salary: Double
- Phone Number: Int
- Address: String

+ Employee()
+ Employee(Id: Int, Name: String, Salary: Double, Phone: Int, Address: String)

+getId(): Int
+setId(Id:int):void

+getName(): String
+setName(Name:string):void

+getSalary(): Double
+setSalary(Salary:double):void

+getPhone(): Int
+setPhone(Phone:int):void

+getAddress(): String
+setAddress (Address:string):void

+toString(): String

My sql Workbench (Queries):-

1. **CREATE SCHEMA `lab`**

2. **use lab**

3. **CREATE TABLE `lab`.`employee` (**

`e_id` INT NOT NULL,

`name` VARCHAR(45) NULL,

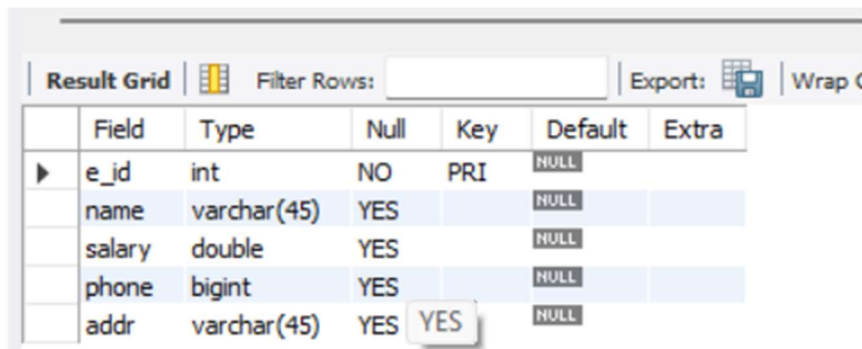
`salary` DOUBLE NULL,

`phone` BIGINT(10) NULL,

`addr` VARCHAR(45) NULL,

PRIMARY KEY (`e_id`));

4. **desc employee**



	Field	Type	Null	Key	Default	Extra
▶	e_id	int	NO	PRI	NULL	
	name	varchar(45)	YES		NULL	
	salary	double	YES		NULL	
	phone	bigint	YES		NULL	
	addr	varchar(45)	YES	YES	NULL	

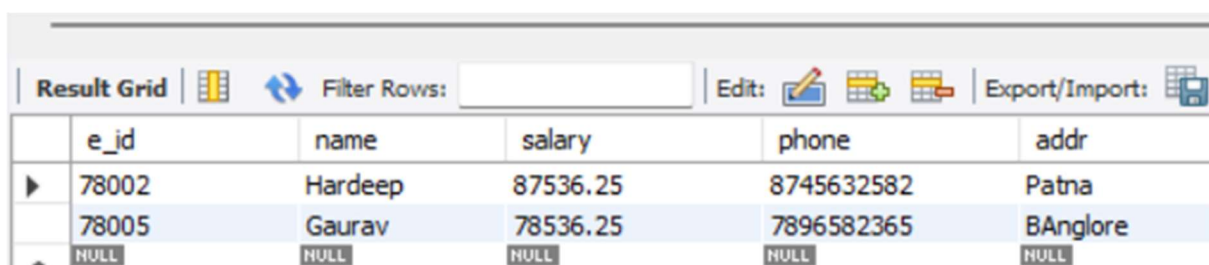
5. **INSERT INTO `lab`.`employee` (`e_id`, `name`, `salary`, `phone`, `addr`)**

VALUES ('78002', 'Hardeep', '87536.25', '8745632582', 'Patna');

INSERT INTO `lab`.`employee` (`e_id`, `name`, `salary`, `phone`, `addr`) VALUES

('78005', 'Gaurav', '78536.25', '7896582365', 'BAnglore');

6. **select * from employee**



	e_id	name	salary	phone	addr
▶	78002	Hardeep	87536.25	8745632582	Patna
	78005	Gaurav	78536.25	7896582365	BAnglore
✱	NULL	NULL	NULL	NULL	NULL

Project in Eclipse:-

Pom.xml :-

```
<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/xsd/maven-
4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <groupId>Employee</groupId>

    <artifactId>Lab_03</artifactId>

    <version>0.0.1-SNAPSHOT</version>

    <packaging>jar</packaging>

    <name>Lab_03</name>

    <url>http://maven.apache.org</url>

    <properties>

        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

    </properties>

    <dependencies>

        <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
        <dependency>

            <groupId>mysql</groupId>

            <artifactId>mysql-connector-java</artifactId>

            <version>8.0.33</version>

        </dependency>

        <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
        <dependency>

            <groupId>org.hibernate</groupId>

            <artifactId>hibernate-core</artifactId>
```

```
<version>5.4.24.Final</version>

</dependency>

<dependency>

  <groupId>junit</groupId>

  <artifactId>junit</artifactId>

  <version>3.8.1</version>

  <scope>test</scope>

</dependency>

</dependencies>

</project>
```

hibernate.cfg.xml :- (Database Connectivity)

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE hibernate-configuration PUBLIC

  "-//Hibernate/Hibernate Configuration DTD 3.0//EN"

  "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

  <session-factory>

    <property name="connection.driver_class">com.mysql.cj.jdbc.Driver</property>

    <property name="connection.url">jdbc:mysql://localhost:3306/lab</property>

    <property name="connection.username">root</property>

    <property name="connection.password">praveer</property>

    <property name="dialect">org.hibernate.dialect.MySQLDialect</property>

    <property name="show_sql">>false</property>

    <property name="hbm2ddl.auto">update</property>

    <mapping resource="Employee/Lab_03/Employee.hbm.xml" />

  </session-factory>

</hibernate-configuration>
```


Employee.hbm.xml :- (configure pojo with main)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<hibernate-mapping>

<class name="Employee.Lab_03.Employee" table="employee">

    <id name="id" type="int" column="e_id">
        <generator class="increment" />
    </id>

    <property name="name">
        <column name="name" />
    </property>

    <property name="salary">
        <column name="salary" />
    </property>

    <property name="phone">
        <column name="phone" />
    </property>

    <property name="addr">
        <column name="addr" />
    </property>

</class>

</hibernate-mapping>
```

Employee.java :- (pojo file)

```
package Employee.Lab_03;

public class Employee {

    private int id;

    private String name;

    private double salary;

    private long phone;

    private String addr;

    public Employee() {

        super();

    }

    public Employee(int id, String name, double salary, long phone, String addr) {

        super();

        this.id = id;

        this.name = name;

        this.salary = salary;

        this.phone = phone;

        this.addr = addr;

    }

    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 double getSalary() {  
    return salary;  
}  
  
public void setSalary(double salary) {  
    this.salary = salary;  
}  
  
public long getPhone() {  
    return phone;  
}  
  
public void setPhone(long phone) {  
    this.phone = phone;  
}  
  
public String getAddr() {  
    return addr;  
}  
  
public void setAddr(String addr) {  
    this.addr = addr;  
}  
  
@Override  
public String toString() {  
    return "Employee [id=" + id + ", name=" + name + ", salary=" + salary + ", phone=" + phone +  
    ", addr=" + addr  
        + "];"  
}  
}}
```

App.java :- (main file)

```
package Employee.Lab_03;

import java.util.List;

import java.util.Scanner;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.cfg.Configuration;


public class App {

    public static void main(String[] args) {

        Configuration config = new Configuration();

        config.configure("hibernate.cfg.xml");

        SessionFactory sf = config.buildSessionFactory();

        Session session = sf.openSession();

        Transaction tx = session.beginTransaction();

        int choice;

        Scanner scanner = new Scanner(System.in);

        do {

            System.out.println("_____");

            System.out.println("---Welcome Employee Management System---");

            System.out.println("-----");

            System.out.println("Choose an option:");

            System.out.println("1. Insert");

            System.out.println("2. Fetch");

            System.out.println("3. Edit or modify the data");
```

```
System.out.println("4. Delete");

System.out.println("5. View full table");

System.out.println("0. Exit");

System.out.print("Enter your choice: ");

choice = scanner.nextInt();

switch (choice) {

case 1: // Insert

    Employee e1 = new Employee();

    System.out.println("Enter name:");

    e1.setName(scanner.next());

    System.out.println("Enter salary:");

    e1.setSalary(scanner.nextDouble());

    System.out.println("Enter phone:");

    e1.setPhone(scanner.nextLong());

    System.out.println("Enter address:");

    e1.setAddr(scanner.next());

    session.save(e1);

    tx.commit();

    System.out.println("Information successfully stored...");

    break;

case 2: // Fetch

    System.out.println("Enter employee ID to fetch:");

    int id = scanner.nextInt();

    Employee fetchedEmployee = session.get(Employee.class, id);

    System.out.println(fetchedEmployee);

    break;

case 3: // Edit or modify

    System.out.println("Enter employee ID to edit:");
```

```
int idToUpdate = scanner.nextInt();

Employee employeeToUpdate = session.get(Employee.class, idToUpdate);

System.out.println("Enter new name:");

employeeToUpdate.setName(scanner.next());

System.out.println("Enter new salary:");

employeeToUpdate.setSalary(scanner.nextDouble());

System.out.println("Enter new phone:");

employeeToUpdate.setPhone(scanner.nextLong());

System.out.println("Enter new address:");

employeeToUpdate.setAddr(scanner.next());

session.saveOrUpdate(employeeToUpdate);

tx.commit();

System.out.println("Information successfully updated...");

break;
```

case 4: // Delete

```
System.out.println("Enter employee ID to delete:");

int idToDelete = scanner.nextInt();

Employee employeeToDelete = session.get(Employee.class, idToDelete);

session.delete(employeeToDelete);

tx.commit();

System.out.println("Information successfully deleted...");

break;
```

case 5: // View full table

```
@SuppressWarnings("unchecked") List<Employee> employees =
session.createQuery("from Employee").list();

for (Employee emp : employees) {

    System.out.println(emp);

}
```

```

        break;

    case 0:

        System.out.println("Goodbye! Have a great day!");

        break;

    default:

        System.out.println("Invalid option!");

    }

} while (choice != 0);

tx.commit();

session.close();

sf.close();

scanner.close();

}

}

```

-: Output :-

```

---Welcome Employee Management System---
-----
Choose an option:
1. Insert
2. Fetch
3. Edit or modify the data
4. Delete
5. View full table
0. Exit
Enter your choice: 1
Enter name:
Niranjan
Enter salary:
78965.25
Enter phone:
7865896541
Enter address:
Hyderabad
Hibernate: select max(e_id) from employee
Hibernate: insert into employee (name, salary, phone, addr, e_id) values (?, ?, ?, ?, ?)
Information successfully stored...

```

Fig-1. Insert employee Information

```

---Welcome Employee Management System---
-----
Choose an option:
1. Insert
2. Fetch
3. Edit or modify the data
4. Delete
5. View full table
0. Exit
Enter your choice: 2
Enter employee ID to fetch:
78005
Hibernate: select employee0_.e_id as e_id1_0_0_, employee0_.name as name2_0_0_, emp
Employee [id=78005, name=Gaurav, salary=78536.25, phone=7896582365, addr=BAnglore]

```

Fig-2. Fetch Information

```

---Welcome Employee Management System---
-----
Choose an option:
1. Insert
2. Fetch
3. Edit or modify the data
4. Delete
5. View full table
0. Exit
Enter your choice: 3
Enter employee ID to edit:
78005
Hibernate: select employee0_.e_id as e_id1_0_0_, employee0_.name as name2_0_0_
Enter new name:
Praveer
Enter new salary:
89025.56
Enter new phone:
9631638269
Enter new address:
Vizag
Hibernate: update employee set name=?, salary=?, phone=?, addr=? where e_id=?
Information successfully updated...

```

Fig-3. Update employee Information


```
---Welcome Employee Management System---
```

```
-----
```

```
Choose an option:
```

1. Insert
2. Fetch
3. Edit or modify the data
4. Delete
5. View full table
0. Exit

```
Enter your choice: 4
```

```
Enter employee ID to delete:
```

```
78006
```

```
Hibernate: select employee0_.e_id as e_id1_0_0_, emp
```

```
Hibernate: delete from employee where e_id=?
```

```
Information successfully deleted...
```

Fig-4. Remove employee Record

```
---Welcome Employee Management System---
```

```
-----
```

```
Choose an option:
```

1. Insert
2. Fetch
3. Edit or modify the data
4. Delete
5. View full table
0. Exit

```
Enter your choice: 5
```

```
Hibernate: select employee0_.e_id as e_id1_0_, employee0_.name as name2_0_, emplo
```

```
Employee [id=78002, name=Hardeep, salary=87536.25, phone=8745632582, addr=Patna]
```

```
Employee [id=78005, name=Praveer, salary=89025.56, phone=9631638269, addr=Vizag]
```

Fig-5. Display all employee

THANK YOU

```
---Welcome Employee Management System---
```

```
-----
```

```
Choose an option:
```

1. Insert
2. Fetch
3. Edit or modify the data
4. Delete
5. View full table
0. Exit

```
Enter your choice: 0
```

```
Goodbye! Have a great day!
```