**WEEK-3**

**Spring Data JPA - Quick Example**

CODE:

**OrmLearnApplication.java:**

package com.cognizant.ormlearn;  
import com.cognizant.ormlearn.model.Country;  
import com.cognizant.ormlearn.service.CountryService;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.context.ApplicationContext;  
import java.util.List;  
  
@SpringBootApplication  
public class OrmLearnApplication {  
 private static final Logger *LOGGER* = LoggerFactory.*getLogger*(OrmLearnApplication.class);  
 private static CountryService *countryService*;  
 public static void main(String[] args) {  
 ApplicationContext context = SpringApplication.*run*(OrmLearnApplication.class, args);  
 *LOGGER*.info("Inside main");  
 *countryService* = context.getBean(CountryService.class);  
 *testGetAllCountries*();  
 }  
 private static void testGetAllCountries() {  
 *LOGGER*.info("Start");  
 List<Country> countries = *countryService*.getAllCountries();  
 *LOGGER*.debug("countries={}", countries);  
 *LOGGER*.info("End");  
 }  
}

**Country.java:**

package com.cognizant.ormlearn.model;  
  
import javax.persistence.Column;  
import javax.persistence.Entity;  
import javax.persistence.Id;  
import javax.persistence.Table;  
  
@Entity  
@Table(name = "country")  
public class Country {  
 @Id  
 @Column(name = "code")  
 private String code;  
 @Column(name = "name")  
 private String name;  
 // Getters and setters  
 public String getCode() {  
 return code;  
 }  
 public void setCode(String code) {  
 this.code = code;  
 }  
 public String getName() {  
 return name;  
 }  
 public void setName(String name) {  
 this.name = name;  
 }  
 // toString  
 @Override  
 public String toString() {  
 return "Country [code=" + code + ", name=" + name + "]";  
 }  
}

**CountryService.java:**

package com.cognizant.ormlearn.service;  
import com.cognizant.ormlearn.model.Country;  
import com.cognizant.ormlearn.repository.CountryRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
import org.springframework.transaction.annotation.Transactional;  
import java.util.List;  
@Service  
public class CountryService {  
 @Autowired  
 private CountryRepository countryRepository;  
 @Transactional  
 public List<Country> getAllCountries() {  
 return countryRepository.findAll();  
 }  
}

**CountryRepository.java:**

package com.cognizant.ormlearn.repository;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
import com.cognizant.ormlearn.model.Country;  
@Repository  
public interface CountryRepository extends JpaRepository<Country, String> {  
}

**applicationProperties.java:**

# Logging configuration  
logging.level.org.springframework=info  
logging.level.com.cognizant=debug  
logging.level.org.hibernate.SQL=trace  
logging.level.org.hibernate.type.descriptor.sql=trace  
logging.pattern.console=%d{dd-MM-yy} %d{HH:mm:ss.SSS} %-20.20thread %5p %-25.25logger{25} %25M %4L %m%n  
  
# Database configuration  
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver  
spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn  
spring.datasource.username=root  
spring.datasource.password=Root@2027  
  
# Hibernate config  
spring.jpa.hibernate.ddl-auto=validate  
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect

**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>com.cognizant</groupId>  
 <artifactId>orm-learn</artifactId>  
 <version>1.0-SNAPSHOT</version>  
 <packaging>jar</packaging>  
 <name>orm-learn</name>  
 <description>Demo project for Spring Data JPA and Hibernate</description>  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>2.7.12</version>  
 <relativePath/>  
 </parent>  
 <properties>  
 <java.version>1.8</java.version>  
 </properties>  
 <dependencies>  
 <!-- Spring Data JPA -->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
 </dependency>  
 <!-- MySQL Connector -->  
 <dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
 <version>8.0.33</version> <!-- Version added --> <scope>runtime</scope>  
 </dependency>  
 <!-- Logging -->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-logging</artifactId>  
 </dependency>  
  
 <!-- DevTools (optional for hot reload) -->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-devtools</artifactId>  
 <scope>runtime</scope>  
 </dependency>  
 </dependencies>  
 <build>  
 <plugins>  
 <!-- Compiler Plugin -->  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-compiler-plugin</artifactId>  
 <version>3.8.1</version>  
 <configuration>  
 <source>1.8</source>  
 <target>1.8</target>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
</project>

**Creating table:**

USE ormlearn;

CREATE TABLE Country (

code VARCHAR(2) PRIMARY KEY,

name VARCHAR(50)

);

DROP TABLE IF EXISTS Country;

INSERT INTO country VALUES ('IN', 'India');

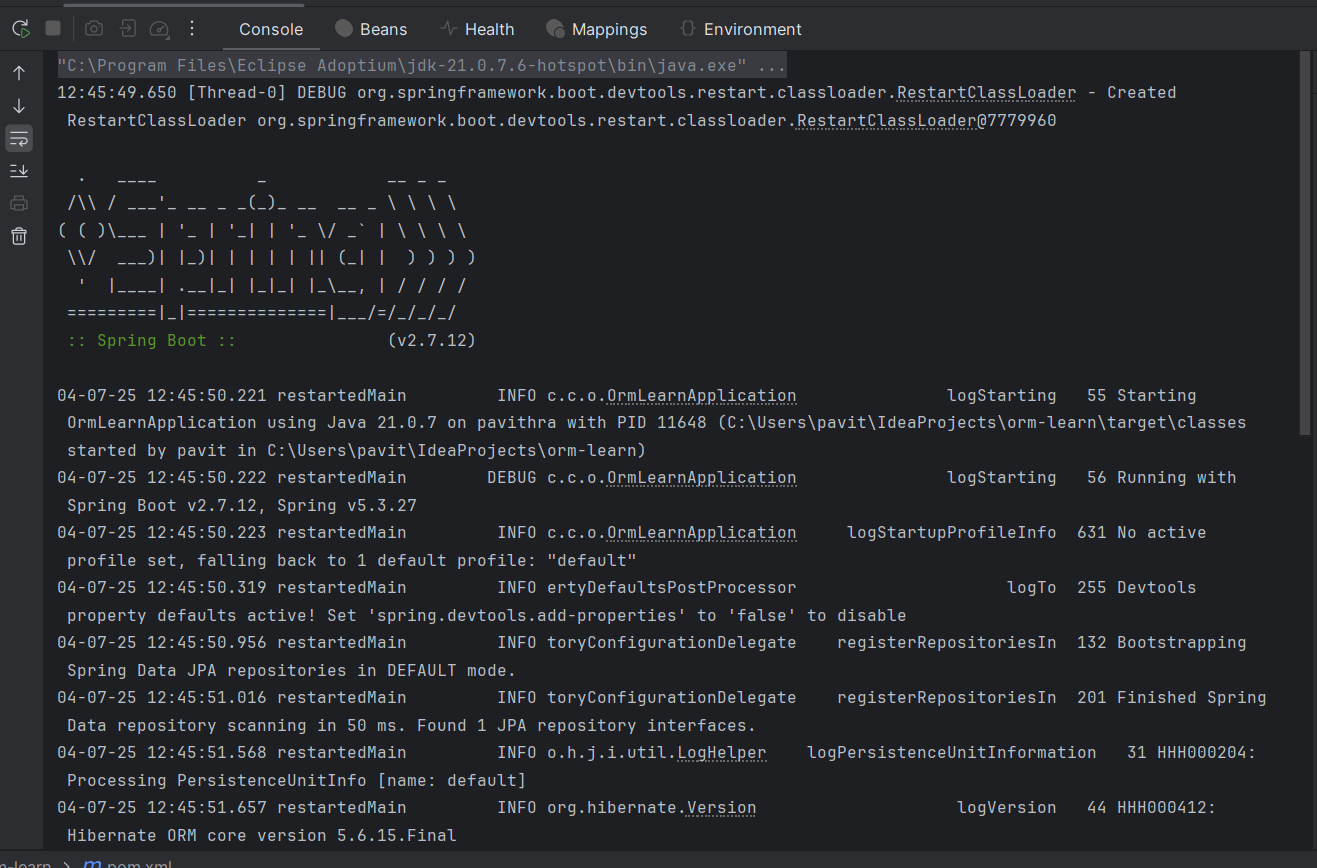
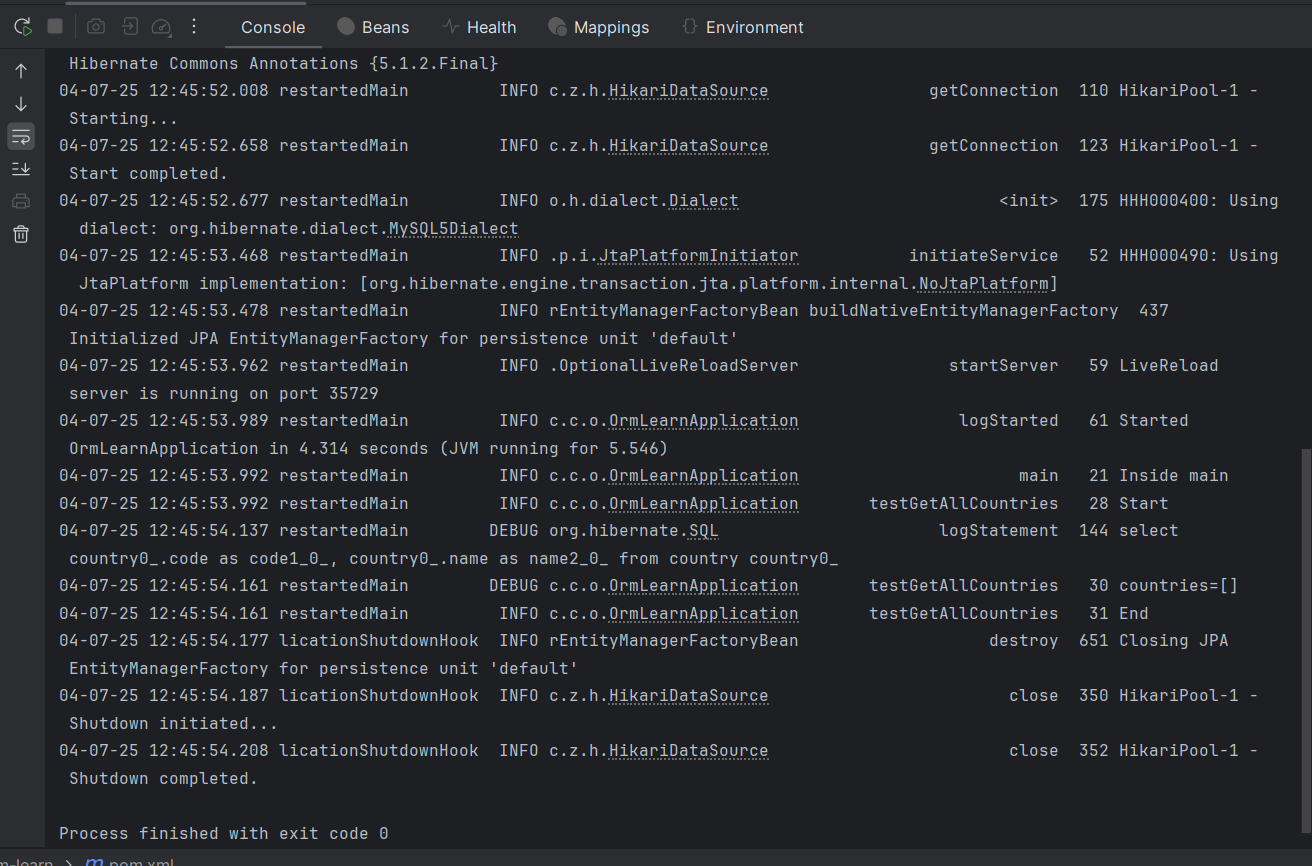
INSERT INTO country VALUES ('US', 'United States of America');

INSERT INTO country VALUES ('UK', 'United Kingdom');

INSERT INTO country VALUES ('AU', 'Australia');

INSERT INTO country VALUES ('CN', 'China');

**OUTPUT:**

****

**Difference between JPA, Hibernate and Spring Data JPA**

**1.Hibernate:**

CODE:

**MainApp.java:**

package org.example.hibernate;  
public class MainApp {  
 public static void main(String[] args) {  
 EmployeeDAO dao = new EmployeeDAO();  
 // Update employee details  
 dao.updateEmployee(1, "John Updated", 40000);  
 // Print all employees  
 dao.getAllEmployees();  
 }  
}

**Employee.java:**

package org.example.hibernate;  
import javax.persistence.\*;  
@Entity  
@Table(name = "employee")  
public class Employee {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private int id;  
 private String name;  
 private String department;  
 private double salary;  
 // Constructors  
 public Employee() {}  
 public Employee(String name, String department, double salary) {  
 this.name = name;  
 this.department = department;  
 this.salary = salary;  
 }  
 // Getters and setters  
 public int getId() {  
 return 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;  
 }  
}

**EmployeeDAO.java:**

package org.example.hibernate;  
import org.hibernate.Session;  
import org.hibernate.Transaction;  
import java.util.List;  
public class EmployeeDAO {  
 public void addEmployee(Employee emp) {  
 try (Session session = HibernateUtil.*getSessionFactory*().openSession()) {  
 Transaction tx = session.beginTransaction();  
 session.save(emp);  
 tx.commit();  
 }  
 }  
 public void updateEmployee(int id, String newName, double newSalary) {  
 try (Session session = HibernateUtil.*getSessionFactory*().openSession()) {  
 Transaction tx = session.beginTransaction();  
 Employee emp = session.get(Employee.class, id);  
 if (emp != null) {  
 emp.setName(newName);  
 emp.setSalary(newSalary);  
 session.update(emp);  
 System.*out*.println("Updated Employee with ID: " + id);  
 } else {  
 System.*out*.println(" Employee not found for ID: " + id);  
 }  
 tx.commit();  
 }  
 }  
 public void getAllEmployees() {  
 try (Session session = HibernateUtil.*getSessionFactory*().openSession()) {  
 List<Employee> employees = session.createQuery("from Employee", Employee.class).list();  
 System.*out*.println("All Employees:");  
 for (Employee emp : employees) {  
 System.*out*.println("ID: " + emp.getId() + ", Name: " + emp.getName() + ", Salary: " + emp.getSalary());  
 }  
 }  
 }  
}

**HibernateUtil.java:**

package org.example.hibernate;  
import org.hibernate.SessionFactory;  
import org.hibernate.cfg.Configuration;  
public class HibernateUtil {  
 private static final SessionFactory *sessionFactory*;  
  
 static {  
 try {  
 *sessionFactory* = new Configuration()  
 .configure("hibernate.cfg.xml")  
 .addAnnotatedClass(Employee.class)  
 .buildSessionFactory();  
 } catch (Throwable ex) {  
 throw new ExceptionInInitializerError(ex);  
 }  
 }  
 public static SessionFactory getSessionFactory() {  
 return *sessionFactory*;  
 }  
}

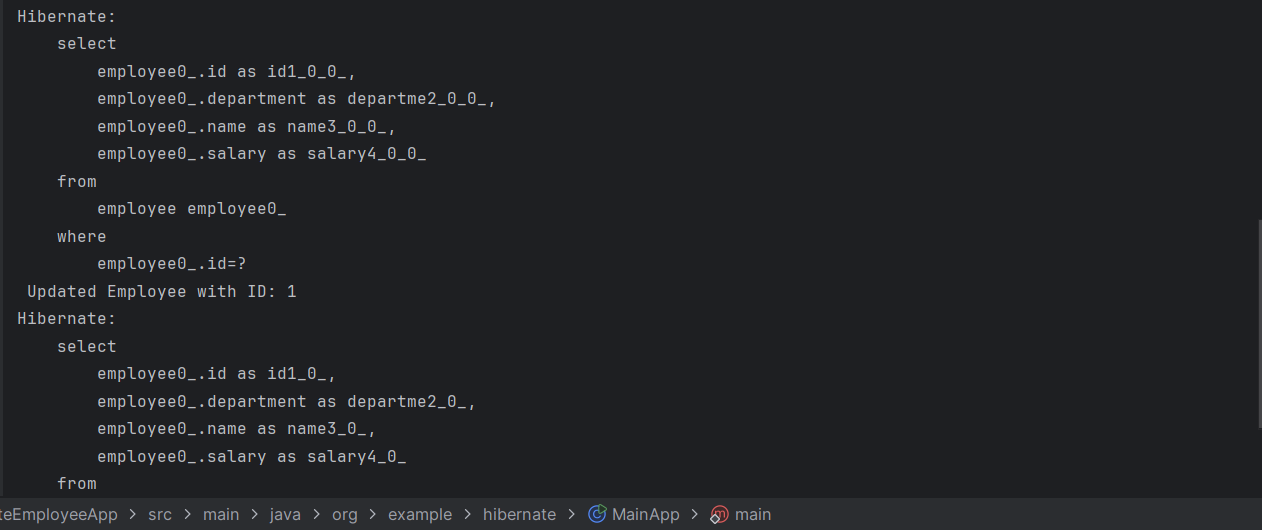
**Hibernatecfg.xml:**

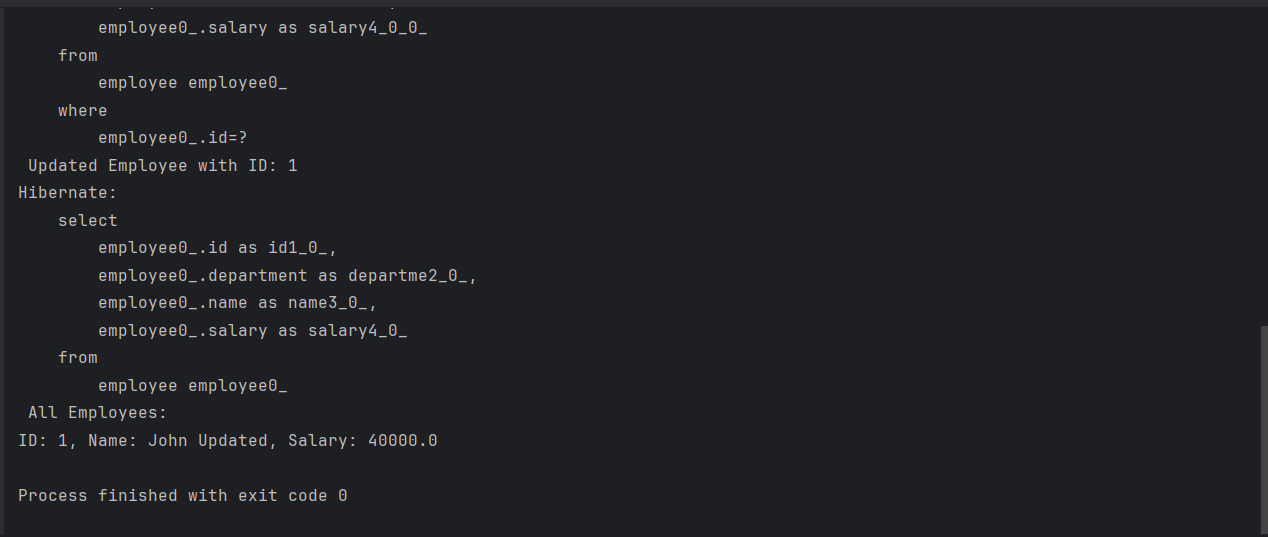
<?xml version='1.0' encoding='utf-8'?>  
<!DOCTYPE hibernate-configuration PUBLIC  
 "-//Hibernate/Hibernate Configuration DTD 3.0//EN"  
 "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">  
<hibernate-configuration>  
 <session-factory>  
 <!-- Database connection settings -->  
 <property name="hibernate.connection.driver\_class">com.mysql.cj.jdbc.Driver</property>  
 <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/omlearn</property>  
 <property name="hibernate.connection.username">root</property>  
 <property name="hibernate.connection.password">Root@2027</property>  
  
 <!-- JDBC connection pool (use the built-in) -->  
 <property name="connection.pool\_size">5</property>  
 <!-- SQL dialect -->  
 <property name="hibernate.dialect">org.hibernate.dialect.MySQL8Dialect</property>  
  
 <!-- Enable Hibernate's automatic session context management -->  
 <property name="current\_session\_context\_class">thread</property>  
  
 <!-- Disable second-level cache -->  
 <property name="cache.provider\_class">org.hibernate.cache.internal.NoCachingRegionFactory</property>  
  
 <!-- Echo all executed SQL to stdout -->  
 <property name="show\_sql">true</property>  
 <property name="format\_sql">true</property>  
  
 <!-- Automatically create or update the schema -->  
 <property name="hbm2ddl.auto">update</property>  
  
 <!-- Mention your entity class mapping -->  
 <mapping class="org.example.hibernate.Employee"/>  
  
 </session-factory>  
</hibernate-configuration>

**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>org.example</groupId>  
 <artifactId>HibernateEmployeeApp</artifactId>  
 <version>1.0-SNAPSHOT</version>  
 <dependencies>  
 <!-- Hibernate Core -->  
 <dependency>  
 <groupId>org.hibernate</groupId>  
 <artifactId>hibernate-core</artifactId>  
 <version>5.6.15.Final</version>  
 </dependency>  
 <!-- MySQL JDBC Connector -->  
 <dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
 <version>8.0.33</version>  
 </dependency>  
 <!-- JPA API -->  
 <dependency>  
 <groupId>javax.persistence</groupId>  
 <artifactId>javax.persistence-api</artifactId>  
 <version>2.2</version>  
 </dependency>  
 </dependencies>  
</project>

**OUTPUT:**

****

****

**2.** **Spring Data JPA:**

CODE:

**MainApp.java:**

package com.example.springjpademo;  
import com.example.springjpademo.entity.Employee;  
import com.example.springjpademo.service.EmployeeService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.boot.CommandLineRunner;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class MainApp implements CommandLineRunner {  
 @Autowired  
 private EmployeeService employeeService;  
 public static void main(String[] args) {  
 SpringApplication.*run*(MainApp.class, args);  
 }  
 @Override  
 public void run(String... args) {  
 System.*out*.println("Inserting employee records...");  
 employeeService.addEmployee(new Employee("John", "HR", 35000));  
 employeeService.addEmployee(new Employee("Alice", "IT", 55000));  
 employeeService.addEmployee(new Employee("Bob", "Finance", 45000));  
 employeeService.addEmployee(new Employee("Diana", "Marketing", 40000));  
 System.*out*.println("Data insertion complete.");  
 }  
}

**Employee.java:**

package com.example.springjpademo.entity;  
import jakarta.persistence.\*;  
@Entity  
@Table(name = "employee")  
public class Employee {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private int id;  
 private String name;  
 private String department;  
 private double salary;  
 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 String getName() {  
 return name;  
 }  
 public String getDepartment() {  
 return department;  
 }  
 public double getSalary() {  
 return salary;  
 }  
 public void setId(int id) {  
 this.id = id;  
 }  
 public void setName(String name) {  
 this.name = name;  
 }  
 public void setDepartment(String department) {  
 this.department = department;  
 }  
 public void setSalary(double salary) {  
 this.salary = salary;  
 }  
}

**EmployeeRepository.java:**

package com.example.springjpademo.repository;  
import com.example.springjpademo.entity.Employee;  
import org.springframework.data.jpa.repository.JpaRepository;  
public interface EmployeeRepository extends JpaRepository<Employee, Integer> {  
}

**EmployeeService.java:**

package com.example.springjpademo.service;  
import com.example.springjpademo.entity.Employee;  
import com.example.springjpademo.repository.EmployeeRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
import jakarta.transaction.Transactional;  
@Service  
public class EmployeeService {  
 @Autowired  
 private EmployeeRepository employeeRepository;  
 @Transactional  
 public void addEmployee(Employee employee) {  
 employeeRepository.save(employee);  
 }  
}

**application.properties:**

# MySQL DB Configuration  
spring.datasource.url=jdbc:mysql://localhost:3306/employeedb  
spring.datasource.username=root  
spring.datasource.password=Root@2027  
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver  
  
# Hibernate Configuration  
spring.jpa.hibernate.ddl-auto=update  
spring.jpa.show-sql=true  
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect

**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>com.example</groupId>  
 <artifactId>springjpademo</artifactId>  
 <version>1.0</version>  
 <packaging>jar</packaging>  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>3.2.4</version>  
 </parent>  
  
 <dependencies>  
 <!-- Spring Boot Starter Data JPA -->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
 </dependency>  
  
 <!-- MySQL Driver -->  
 <dependency>  
 <groupId>com.mysql</groupId>  
 <artifactId>mysql-connector-j</artifactId>  
 <scope>runtime</scope>  
 </dependency>  
  
 <!-- Spring Boot Starter -->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter</artifactId>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
</project>

**OUTPUT:**

