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

**//Employee.java**

package com.example;

import javax.persistence.\*;

@Entity

@Table(name = "employees")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "id")

private Integer id;

@Column(name = "first\_name", nullable = false)

private String firstName;

@Column(name = "last\_name", nullable = false)

private String lastName;

@Column(name = "email", unique = true, nullable = false)

private String email;

@Column(name = "salary")

private Double salary;

// Default constructor

public Employee() {}

// Constructor with parameters

public Employee(String firstName, String lastName, String email, Double salary) {

this.firstName = firstName;

this.lastName = lastName;

this.email = email;

this.salary = salary;

}

// Getters and Setters

public Integer getId() {

return id;

}

public void setId(Integer id) {

this.id = id;

}

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public Double getSalary() {

return salary;

}

public void setSalary(Double salary) {

this.salary = salary;

}

@Override

public String toString() {

return "Employee{" +

"id=" + id +

", firstName='" + firstName + '\'' +

", lastName='" + lastName + '\'' +

", email='" + email + '\'' +

", salary=" + salary +

'}';

}

}

**//EmployeeDAO.java**

package com.example;

import org.hibernate.HibernateException;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.cfg.Configuration;

import java.util.List;

public class EmployeeDAO {

private static SessionFactory factory;

static {

try {

factory = new Configuration().configure().buildSessionFactory();

} catch (Throwable ex) {

System.err.println("Failed to create sessionFactory object." + ex);

throw new ExceptionInInitializerError(ex);

}

}

/\* Method to CREATE an employee in the database \*/

public Integer addEmployee(Employee employee) {

Session session = factory.openSession();

Transaction tx = null;

Integer employeeID = null;

try {

tx = session.beginTransaction();

employeeID = (Integer) session.save(employee);

tx.commit();

} catch (HibernateException e) {

if (tx != null) tx.rollback();

e.printStackTrace();

} finally {

session.close();

}

return employeeID;

}

/\* Method to READ all employees from the database \*/

public List<Employee> listEmployees() {

Session session = factory.openSession();

Transaction tx = null;

List<Employee> employees = null;

try {

tx = session.beginTransaction();

employees = session.createQuery("FROM Employee", Employee.class).list();

tx.commit();

} catch (HibernateException e) {

if (tx != null) tx.rollback();

e.printStackTrace();

} finally {

session.close();

}

return employees;

}

/\* Method to UPDATE employee in the database \*/

public void updateEmployee(Integer employeeID, Double salary) {

Session session = factory.openSession();

Transaction tx = null;

try {

tx = session.beginTransaction();

Employee employee = session.get(Employee.class, employeeID);

employee.setSalary(salary);

session.update(employee);

tx.commit();

} catch (HibernateException e) {

if (tx != null) tx.rollback();

e.printStackTrace();

} finally {

session.close();

}

}

/\* Method to DELETE an employee from the database \*/

public void deleteEmployee(Integer employeeID) {

Session session = factory.openSession();

Transaction tx = null;

try {

tx = session.beginTransaction();

Employee employee = session.get(Employee.class, employeeID);

session.delete(employee);

tx.commit();

} catch (HibernateException e) {

if (tx != null) tx.rollback();

e.printStackTrace();

} finally {

session.close();

}

}

/\* Method to close the SessionFactory \*/

public static void shutdown() {

factory.close();

}

}

**//HibernateExample.java**

package com.example;

import java.util.List;

public class HibernateExample {

public static void main(String[] args) {

EmployeeDAO employeeDAO = new EmployeeDAO();

System.out.println("=== Hibernate Example ===");

System.out.println("Demonstrating manual session and transaction management\n");

// Create employees

System.out.println("1. Creating employees...");

Integer emp1Id = employeeDAO.addEmployee(new Employee("John", "Doe", "john.doe@example.com", 50000.0));

Integer emp2Id = employeeDAO.addEmployee(new Employee("Jane", "Smith", "jane.smith@example.com", 60000.0));

Integer emp3Id = employeeDAO.addEmployee(new Employee("Bob", "Johnson", "bob.johnson@example.com", 55000.0));

System.out.println("Created employee with ID: " + emp1Id);

System.out.println("Created employee with ID: " + emp2Id);

System.out.println("Created employee with ID: " + emp3Id);

// List all employees

System.out.println("\n2. Listing all employees:");

List<Employee> employees = employeeDAO.listEmployees();

for (Employee emp : employees) {

System.out.println(emp);

}

// Update an employee

System.out.println("\n3. Updating employee salary...");

employeeDAO.updateEmployee(emp1Id, 65000.0);

// List employees after update

System.out.println("\n4. Listing employees after update:");

employees = employeeDAO.listEmployees();

for (Employee emp : employees) {

System.out.println(emp);

}

// Delete an employee

System.out.println("\n5. Deleting employee...");

employeeDAO.deleteEmployee(emp2Id);

// List employees after deletion

System.out.println("\n6. Listing employees after deletion:");

employees = employeeDAO.listEmployees();

for (Employee emp : employees) {

System.out.println(emp);

}

System.out.println("\n=== Hibernate Example Completed ===");

System.out.println("Notice the manual session and transaction management required!");

// Shutdown

EmployeeDAO.shutdown();

}

}

**//pom.xml**

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

<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>hibernate-example</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>jar</packaging>

<properties>

<maven.compiler.source>11</maven.compiler.source>

<maven.compiler.target>11</maven.compiler.target>

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

<hibernate.version>5.6.15.Final</hibernate.version>

</properties>

<dependencies>

<!-- Hibernate Core -->

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>${hibernate.version}</version>

</dependency>

<!-- H2 Database -->

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<version>2.1.214</version>

</dependency>

<!-- JUnit for testing -->

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter</artifactId>

<version>5.9.0</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.8.1</version>

<configuration>

<source>11</source>

<target>11</target>

</configuration>

</plugin>

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>exec-maven-plugin</artifactId>

<version>3.1.0</version>

<configuration>

<mainClass>com.example.HibernateExample</mainClass>

</configuration>

</plugin>

</plugins>

</build>

</project>

**//EmployeeRepository.java**

package com.example;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

@Repository

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

// Custom query methods can be added here

// Spring Data JPA will automatically implement these methods

// Find employees by first name

java.util.List<Employee> findByFirstName(String firstName);

// Find employees by email

Employee findByEmail(String email);

// Find employees with salary greater than specified amount

java.util.List<Employee> findBySalaryGreaterThan(Double salary);

// Find employees by first name and last name

java.util.List<Employee> findByFirstNameAndLastName(String firstName, String lastName);

}

**//EmployeeService.java**

package com.example;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import java.util.List;

import java.util.Optional;

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

@Transactional(readOnly = true)

public List<Employee> getAllEmployees() {

return employeeRepository.findAll();

}

@Transactional(readOnly = true)

public Optional<Employee> getEmployeeById(Integer id) {

return employeeRepository.findById(id);

}

@Transactional

public Employee updateEmployee(Employee employee) {

return employeeRepository.save(employee);

}

@Transactional

public void deleteEmployee(Integer id) {

employeeRepository.deleteById(id);

}

@Transactional(readOnly = true)

public List<Employee> getEmployeesByFirstName(String firstName) {

return employeeRepository.findByFirstName(firstName);

}

@Transactional(readOnly = true)

public Employee getEmployeeByEmail(String email) {

return employeeRepository.findByEmail(email);

}

@Transactional(readOnly = true)

public List<Employee> getEmployeesWithSalaryGreaterThan(Double salary) {

return employeeRepository.findBySalaryGreaterThan(salary);

}

}

**//SpringDataJpaApplication.java**

package com.example;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import java.util.List;

@SpringBootApplication

public class SpringDataJpaApplication implements CommandLineRunner {

@Autowired

private EmployeeService employeeService;

public static void main(String[] args) {

SpringApplication.run(SpringDataJpaApplication.class, args);

}

@Override

public void run(String... args) throws Exception {

System.out.println("=== Spring Data JPA Example ===");

System.out.println("Demonstrating automatic transaction and session management\n");

// Create employees

System.out.println("1. Creating employees...");

Employee emp1 = new Employee("John", "Doe", "john.doe@example.com", 50000.0);

Employee emp2 = new Employee("Jane", "Smith", "jane.smith@example.com", 60000.0);

Employee emp3 = new Employee("Bob", "Johnson", "bob.johnson@example.com", 55000.0);

employeeService.addEmployee(emp1);

employeeService.addEmployee(emp2);

employeeService.addEmployee(emp3);

System.out.println("Employees created successfully!");

// List all employees

System.out.println("\n2. Listing all employees:");

List<Employee> employees = employeeService.getAllEmployees();

employees.forEach(System.out::println);

// Update an employee

System.out.println("\n3. Updating employee salary...");

if (!employees.isEmpty()) {

Employee firstEmployee = employees.get(0);

firstEmployee.setSalary(65000.0);

employeeService.updateEmployee(firstEmployee);

System.out.println("Updated employee: " + firstEmployee);

}

// List employees after update

System.out.println("\n4. Listing employees after update:");

employees = employeeService.getAllEmployees();

employees.forEach(System.out::println);

// Delete an employee

System.out.println("\n5. Deleting employee...");

if (employees.size() > 1) {

Employee employeeToDelete = employees.get(1);

employeeService.deleteEmployee(employeeToDelete.getId());

System.out.println("Deleted employee with ID: " + employeeToDelete.getId());

}

// List employees after deletion

System.out.println("\n6. Listing employees after deletion:");

employees = employeeService.getAllEmployees();

employees.forEach(System.out::println);

// Demonstrate custom queries

System.out.println("\n7. Demonstrating custom queries:");

List<Employee> highSalaryEmployees = employeeService.getEmployeesWithSalaryGreaterThan(50000.0);

System.out.println("Employees with salary > 50000:");

highSalaryEmployees.forEach(System.out::println);

System.out.println("\n=== Spring Data JPA Example Completed ===");

System.out.println("Notice how Spring Data JPA handled transactions and sessions automatically!");

}

}

**OUTPUT:**

