**Demonstrate writing Hibernate Query Language and Native Query**

**Application.java**

package com.example.countryapp;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class CountryAppApplication {

public static void main(String[] args) {

SpringApplication.run(CountryAppApplication.class, args);

}

}

**Department.java**

package com.example.countryapp.model;

import jakarta.persistence.\*;

import java.util.List;

@Entity

public class Department {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

@OneToMany(mappedBy = "department", fetch = FetchType.LAZY)

private List<Employee> employees;

public Long getId() { return id; }

public void setId(Long id) { this.id = id; }

public String getName() { return name; }

public void setName(String name) { this.name = name; }

public List<Employee> getEmployees() { return employees; }

public void setEmployees(List<Employee> employees) { this.employees = employees; }

}

**Employee.java**

package com.example.countryapp.model;

import jakarta.persistence.\*;

@Entity

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private Double salary;

@ManyToOne

@JoinColumn(name = "department\_id")

private Department department;

public Long getId() { return id; }

public void setId(Long 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 Department getDepartment() { return department; }

public void setDepartment(Department department) { this.department = department; }

}

**EmployeeRepository.java**

package com.example.countryapp.repository;

import com.example.countryapp.model.Employee;

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

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

import org.springframework.data.repository.query.Param;

import java.util.List;

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

@Query("SELECT e FROM Employee e JOIN FETCH e.department")

List<Employee> findAllWithDepartments();

@Query("SELECT e FROM Employee e WHERE e.department.name = :name")

List<Employee> findByDepartmentName(@Param("name") String name);

@Query("SELECT COUNT(e) FROM Employee e WHERE e.department.name = :name")

Long countByDepartmentName(@Param("name") String name);

@Query(value = "SELECT name FROM employee", nativeQuery = true)

List<String> findAllEmployeeNamesNative();

}

**DataLoader.java**

package com.example.countryapp.service;

import com.example.countryapp.model.Department;

import com.example.countryapp.model.Employee;

import com.example.countryapp.repository.EmployeeRepository;

import jakarta.persistence.EntityManager;

import jakarta.transaction.Transactional;

import org.springframework.boot.CommandLineRunner;

import org.springframework.stereotype.Component;

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

import java.util.List;

@Component

public class DataLoader implements CommandLineRunner {

@Autowired

private EmployeeRepository employeeRepository;

@Autowired

private EntityManager entityManager;

@Transactional

@Override

public void run(String... args) {

System.out.println("Start");

Department dept = new Department();

dept.setName("HR");

entityManager.persist(dept);

Employee e1 = new Employee();

e1.setName("Alice");

e1.setSalary(50000.0);

e1.setDepartment(dept);

entityManager.persist(e1);

Employee e2 = new Employee();

e2.setName("Bob");

e2.setSalary(60000.0);

e2.setDepartment(dept);

entityManager.persist(e2);

System.out.println("--- Fetch with HQL join fetch ---");

List<Employee> allEmployees = employeeRepository.findAllWithDepartments();

allEmployees.forEach(e -> System.out.println(e.getName() + " - " + e.getDepartment().getName()));

System.out.println("--- HQL: By department name ---");

List<Employee> hrEmployees = employeeRepository.findByDepartmentName("HR");

hrEmployees.forEach(e -> System.out.println(e.getName()));

System.out.println("--- HQL: Count ---");

Long count = employeeRepository.countByDepartmentName("HR");

System.out.println("Employee count in HR: " + count);

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

List<String> names = employeeRepository.findAllEmployeeNamesNative();

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

System.out.println("End");

}

}

**applications.properties**

spring.datasource.url=jdbc:mysql://localhost:3306/yourdb

spring.datasource.username=root

spring.datasource.password=Sajitha\_$2004

spring.jpa.hibernate.ddl-auto=create

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.format\_sql=true











