**WEEK 3**

**EXERCISE 5**

**Employee Management System - Defining Query Methods**

**1. Defining Query Methods:**

**a. Using Keywords in Method Names**

* **Purpose:** Create custom query methods by simply naming them according to the entity fields.

**Examples of Derived Query Methods:**

**EmployeeRepository Code:**

package com.example.employeemanagementsystem.repository;

import com.example.employeemanagementsystem.entity.Employee;

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

import org.springframework.stereotype.Repository;

import java.util.List;

@Repository

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

// Find employees by department name

List<Employee> findByDepartmentName(String departmentName);

// Find employees whose names contain a specific string

List<Employee> findByNameContaining(String keyword);

// Find employees with email ending in a specific domain

List<Employee> findByEmailEndingWith(String domain);

}

**DepartmentRepository Code:**

package com.example.employeemanagementsystem.repository;

import com.example.employeemanagementsystem.entity.Department;

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

import org.springframework.stereotype.Repository;

@Repository

public interface DepartmentRepository extends JpaRepository<Department, Long> {

// Find departments by name ignoring case

Department findByNameIgnoreCase(String name);

// Find all departments that have employees

List<Department> findByEmployeesIsNotNull();

}

**b. Implementing Custom Query Methods with @Query Annotation**

* **Purpose:** Use the @Query annotation to define more complex queries directly in the repository interface.

**Examples of Custom Queries:**

**EmployeeRepository Code:**

package com.example.employeemanagementsystem.repository;

import com.example.employeemanagementsystem.entity.Employee;

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

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

import org.springframework.stereotype.Repository;

import java.util.List;

@Repository

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

// Custom query to find employees by department name

@Query("SELECT e FROM Employee e WHERE e.department.name = ?1")

List<Employee> findEmployeesByDepartmentName(String departmentName);

// Custom query to find employees with a specific email domain

@Query("SELECT e FROM Employee e WHERE e.email LIKE %?1")

List<Employee> findEmployeesByEmailDomain(String domain);

}

**2. Named Queries:**

**a. Defining Named Queries**

* **Purpose:** Define reusable, named queries in the entity classes using @NamedQuery or @NamedQueries annotations.

**Examples of Named Queries:**

**Employee Entity Code:**

package com.example.employeemanagementsystem.entity;

import lombok.Data;

import javax.persistence.\*;

@Entity

@Table(name = "employees")

@Data

@NamedQueries({

@NamedQuery(name = "Employee.findByDepartmentName",

query = "SELECT e FROM Employee e WHERE e.department.name = :name"),

@NamedQuery(name = "Employee.findByEmailDomain",

query = "SELECT e FROM Employee e WHERE e.email LIKE :domain")

})

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String email;

@ManyToOne

@JoinColumn(name = "department\_id")

private Department department;

}

**b. Executing Named Queries**

* **Purpose:** Execute the named queries defined in the entity classes via the repository or using an EntityManager.

**Example of Executing a Named Query in a Repository***:*

**EmployeeRepository Code:**

package com.example.employeemanagementsystem.repository;

import com.example.employeemanagementsystem.entity.Employee;

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

import org.springframework.stereotype.Repository;

import java.util.List;

@Repository

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

// Method to execute the named query defined in the Employee entity

List<Employee> findByDepartmentName(String name);

// Method to execute the named query defined in the Employee entity

List<Employee> findByEmailDomain(String domain);

}