**Exercise 5: Employee Management System - Defining Query Methods**

**Business Scenario:**

Enhance your repository to support custom queries.

**Instructions:**

1. **Defining Query Methods:**
   * Use keywords in method names to create custom query methods.
   * Implement custom query methods using the **@Query** annotation.
2. **Named Queries:**
   * Define and execute named queries with **@NamedQuery** and **@NamedQueries**.

**Solution**

**EmployeeRepository.java**

package com.employee.employeemanagement.repository;

import com.employee.employeemanagement.entity.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> {

// Derived query methods

List<Employee> findByName(String name);

List<Employee> findByEmail(String email);

List<Employee> findByDepartmentName(String departmentName);

// Custom query using @Query annotation

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

List<Employee> findByEmailAndName(@Param("email") String email, @Param("name") String name);

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

List<Employee> findByDepartment(@Param("departmentName") String departmentName);

}

**Employee.java**

package com.employee.employeemanagement.entity;

import jakarta.persistence.\*;

@Entity

@Table(name = "employees")

@NamedQueries({

@NamedQuery(name = "Employee.findByName",

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

@NamedQuery(name = "Employee.findByEmail",

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

@NamedQuery(name = "Employee.findByDepartmentName",

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

})

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;

}

**EmployeeService.java**

import jakarta.persistence.EntityManager;

import jakarta.persistence.PersistenceContext;

import jakarta.persistence.TypedQuery;

import org.springframework.stereotype.Service;

@Service

public class EmployeeService {

@PersistenceContext

private EntityManager entityManager;

public List<Employee> getEmployeesByName(String name) {

TypedQuery<Employee> query = entityManager.createNamedQuery("Employee.findByName", Employee.class);

query.setParameter("name", name);

return query.getResultList();

}

}

**Main.java**

package com.employee.employeemanagement;

import com.employee.employeemanagement.entity.Employee;

import com.employee.employeemanagement.repository.EmployeeRepository;

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.stereotype.Component;

import java.util.List;

@Component

public class Main implements CommandLineRunner {

@Autowired

private EmployeeRepository employeeRepository;

@Override

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

List<Employee> employeesByName = employeeRepository.findByName("John Doe");

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

List<Employee> employeesByEmail = employeeRepository.findByEmail("john.doe@example.com");

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

List<Employee> employeesByDepartment = employeeRepository.findByDepartment("IT");

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

List<Employee> employeesByEmailAndName = employeeRepository.findByEmailAndName("john.doe@example.com", "John Doe");

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

}

}