**Exercise 8: Employee Management System - Creating Projections**

**Business Scenario:**

**Create projections to fetch specific data subsets from the employee and department entities.**

**Instructions:**

1. **Projections:**
   * **Define interface-based and class-based projections.**
   * **Use @Value and constructor expressions to control the fetched data.**

**Solution**

**EmployeeProjection.java**

package com.employee.employeemanagement.projection;

public interface EmployeeProjection {

String getName();

String getEmail();

String getDepartmentName();

}

**EmployeeDTO.java**

package com.employee.employeemanagement.dto;

public class EmployeeDTO {

private String name;

private String email;

private String departmentName;

public EmployeeDTO(String name, String email, String departmentName) {

this.name = name;

this.email = email;

this.departmentName = departmentName;

}

// Getters and setters (optional if using public fields)

public String getName() {

return name;

}

public String getEmail() {

return email;

}

public String getDepartmentName() {

return departmentName;

}

@Override

public String toString() {

return "EmployeeDTO{" +

"name='" + name + '\'' +

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

", departmentName='" + departmentName + '\'' +

'}';

}

}

**EmployeeRepository.java**

package com.employee.employeemanagement.repository;

import com.employee.employeemanagement.dto.EmployeeDTO;

import com.employee.employeemanagement.entity.Employee;

import com.employee.employeemanagement.projection.EmployeeProjection;

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> {

// Interface-based projection

List<EmployeeProjection> findByDepartmentName(String departmentName);

// Class-based projection with @Query

@Query("SELECT new com.employee.employeemanagement.dto.EmployeeDTO(e.name, e.email, e.department.name) " +

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

List<EmployeeDTO> findEmployeeDTOByDepartmentName(@Param("departmentName") String departmentName);

}

**EmployeeProjection.java**

package com.employee.employeemanagement.projection;

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

public interface EmployeeProjection {

String getName();

String getEmail();

@Value("#{target.department.name}")

String getDepartmentName();

}

**Main.java**

package com.employee.employeemanagement;

import com.employee.employeemanagement.dto.EmployeeDTO;

import com.employee.employeemanagement.projection.EmployeeProjection;

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 {

// Interface-based projection

List<EmployeeProjection> employeesProjection = employeeRepository.findByDepartmentName("IT");

employeesProjection.forEach(employee -> {

System.out.println("Name: " + employee.getName());

System.out.println("Email: " + employee.getEmail());

System.out.println("Department: " + employee.getDepartmentName());

});

// Class-based projection

List<EmployeeDTO> employeesDTO = employeeRepository.findEmployeeDTOByDepartmentName("IT");

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

}

}