### 1. Interface-Based Projections

Interface-based projections are a powerful feature of Spring Data JPA that allows you to define an interface with getter methods for the fields you want to fetch. This approach is straightforward and useful for simple use cases.

#### 1.1 Define Interface-Based Projections

**EmployeeProjection.java:**

package com.example.employeemanagementsystem.projection;

public interface EmployeeProjection {

Long getId();

String getName();

String getEmail();

// You can include other fields as needed

}

**DepartmentProjection.java:**

package com.example.employeemanagementsystem.projection;

public interface DepartmentProjection {

Long getId();

String getName();

}

#### 1.2 Use Projections in Repository Methods

Modify the repository methods to return projections instead of entities.

**EmployeeRepository.java:**

package com.example.employeemanagementsystem.repository;

import com.example.employeemanagementsystem.model.Employee;

import com.example.employeemanagementsystem.projection.EmployeeProjection;

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

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

import org.springframework.data.domain.Page;

import org.springframework.data.domain.Pageable;

import org.springframework.stereotype.Repository;

import java.util.List;

@Repository

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

// Using interface projection

@Query("SELECT e FROM Employee e WHERE e.name LIKE %:namePart%")

Page<EmployeeProjection> findEmployeeProjectionsByNameContaining(String namePart, Pageable pageable);

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

Page<EmployeeProjection> findEmployeeProjectionsByDepartmentId(Long departmentId, Pageable pageable);

}

**DepartmentRepository.java:**

package com.example.employeemanagementsystem.repository;

import com.example.employeemanagementsystem.model.Department;

import com.example.employeemanagementsystem.projection.DepartmentProjection;

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

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

import org.springframework.stereotype.Repository;

@Repository

public interface DepartmentRepository extends JpaRepository<Department, Long> {

@Query("SELECT d FROM Department d")

List<DepartmentProjection> findAllDepartmentProjections();

}

### 2. Class-Based Projections

Class-based projections provide more control and are useful for more complex projections. They are defined as classes with the required fields and constructors.

#### 2.1 Define Class-Based Projections

**EmployeeDTO.java:**

package com.example.employeemanagementsystem.dto;

public class EmployeeDTO {

private Long id;

private String name;

private String email;

// Constructor with parameters for initializing the fields

public EmployeeDTO(Long id, String name, String email) {

this.id = id;

this.name = name;

this.email = email;

}

// Getters and setters

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 String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

}

**DepartmentDTO.java:**

package com.example.employeemanagementsystem.dto;

public class DepartmentDTO {

private Long id;

private String name;

// Constructor with parameters for initializing the fields

public DepartmentDTO(Long id, String name) {

this.id = id;

this.name = name;

}

// Getters and setters

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;

}

}

#### 2.2 Use Class-Based Projections in Repository Methods

Modify the repository methods to return DTOs instead of entities.

**EmployeeRepository.java:**

package com.example.employeemanagementsystem.repository;

import com.example.employeemanagementsystem.dto.EmployeeDTO;

import com.example.employeemanagementsystem.model.Employee;

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

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

import org.springframework.data.domain.Page;

import org.springframework.data.domain.Pageable;

import org.springframework.stereotype.Repository;

@Repository

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

// Using class-based projection

@Query("SELECT new com.example.employeemanagementsystem.dto.EmployeeDTO(e.id, e.name, e.email) FROM Employee e WHERE e.name LIKE %:namePart%")

Page<EmployeeDTO> findEmployeeDTOsByNameContaining(String namePart, Pageable pageable);

@Query("SELECT new com.example.employeemanagementsystem.dto.EmployeeDTO(e.id, e.name, e.email) FROM Employee e WHERE e.department.id = :departmentId")

Page<EmployeeDTO> findEmployeeDTOsByDepartmentId(Long departmentId, Pageable pageable);

}

**DepartmentRepository.java:**

package com.example.employeemanagementsystem.repository;

import com.example.employeemanagementsystem.dto.DepartmentDTO;

import com.example.employeemanagementsystem.model.Department;

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 DepartmentRepository extends JpaRepository<Department, Long> {

@Query("SELECT new com.example.employeemanagementsystem.dto.DepartmentDTO(d.id, d.name) FROM Department d")

List<DepartmentDTO> findAllDepartmentDTOs();

}