### 1. Implementing Pagination

Pagination allows you to break down large sets of data into smaller chunks or pages. Spring Data JPA makes this easy with the Page and Pageable interfaces.

#### 1.1 Update EmployeeRepository to Support Pagination

Modify your repository to return a Page of employees instead of a List.

**EmployeeRepository.java**

package com.example.employeemanagementsystem.repository;

import com.example.employeemanagementsystem.model.Employee;

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

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> {

// Find employees by name with pagination

Page<Employee> findByNameContaining(String namePart, Pageable pageable);

// Find employees by department's ID with pagination

Page<Employee> findByDepartmentId(Long departmentId, Pageable pageable);

}

* findByNameContaining(String namePart, Pageable pageable): Finds employees with names containing a specific substring and returns a Page of results.
* findByDepartmentId(Long departmentId, Pageable pageable): Finds employees in a specific department and returns a Page of results.

#### 1.2 Implement Pagination in EmployeeController

Update the controller to handle pagination using Pageable.

**EmployeeController.java**

package com.example.employeemanagementsystem.controller;

import com.example.employeemanagementsystem.model.Employee;

import com.example.employeemanagementsystem.repository.EmployeeRepository;

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

import org.springframework.data.domain.Page;

import org.springframework.data.domain.Pageable;

import org.springframework.web.bind.annotation.\*;

@RestController

@RequestMapping("/employees")

public class EmployeeController {

@Autowired

private EmployeeRepository employeeRepository;

// Endpoint for paginated and optionally sorted employee search

@GetMapping

public Page<Employee> getAllEmployees(

@RequestParam(required = false) String namePart,

@RequestParam(required = false) Long departmentId,

Pageable pageable) {

if (namePart != null) {

return employeeRepository.findByNameContaining(namePart, pageable);

} else if (departmentId != null) {

return employeeRepository.findByDepartmentId(departmentId, pageable);

} else {

return employeeRepository.findAll(pageable);

}

}

// Other CRUD methods remain unchanged

}

* Pageable: Automatically handles pagination and sorting based on query parameters.

### 2. Implementing Sorting

Sorting allows you to order the results based on one or more fields.

#### 2.1 Add Sorting to EmployeeRepository

Sorting is supported out of the box with Pageable, so you don't need to make additional changes to the repository for basic sorting.

#### 2.2 Combine Pagination and Sorting in EmployeeController

The Pageable parameter in Spring MVC automatically supports sorting. You can pass sort parameters in the request URL.

**Example Request URL with Sorting and Pagination:**

GET /employees?page=0&size=10&sort=name,asc

* page: Page number (0-based).
* size: Number of items per page.
* sort: Field and direction (e.g., name,asc for ascending order by name).

**EmployeeController.java (Updated)**

package com.example.employeemanagementsystem.controller;

import com.example.employeemanagementsystem.model.Employee;

import com.example.employeemanagementsystem.repository.EmployeeRepository;

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

import org.springframework.data.domain.Page;

import org.springframework.data.domain.Pageable;

import org.springframework.web.bind.annotation.\*;

@RestController

@RequestMapping("/employees")

public class EmployeeController {

@Autowired

private EmployeeRepository employeeRepository;

// Endpoint for paginated and optionally sorted employee search

@GetMapping

public Page<Employee> getAllEmployees(

@RequestParam(required = false) String namePart,

@RequestParam(required = false) Long departmentId,

Pageable pageable) {

if (namePart != null) {

return employeeRepository.findByNameContaining(namePart, pageable);

} else if (departmentId != null) {

return employeeRepository.findByDepartmentId(departmentId, pageable);

} else {

return employeeRepository.findAll(pageable);

}

}

// Other CRUD methods remain unchanged

}