**WEEK 3**

**EXERCISE 6**

**Employee Management System - Implementing Pagination and Sorting**

**1. Pagination:**

**a. Implementing Pagination for Employee List**

* **Purpose:** Use Page and Pageable to retrieve employee records in a paginated manner, allowing clients to fetch data in chunks rather than all at once.

**EmployeeRepository Code:**

package com.example.employeemanagementsystem.repository;

import com.example.employeemanagementsystem.entity.Employee;

import org.springframework.data.domain.Page;

import org.springframework.data.domain.Pageable;

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

import org.springframework.stereotype.Repository;

@Repository

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

// Method to fetch paginated list of employees

Page<Employee> findAll(Pageable pageable);

}

**EmployeeController Code:**

package com.example.employeemanagementsystem.controller;

import com.example.employeemanagementsystem.entity.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.PageRequest;

import org.springframework.data.domain.Pageable;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.RestController;

@RestController

@RequestMapping("/api/employees")

public class EmployeeController {

@Autowired

private EmployeeRepository employeeRepository;

// Get paginated list of employees

@GetMapping("/paged")

public Page<Employee> getEmployeesPaged(

@RequestParam(defaultValue = "0") int page,

@RequestParam(defaultValue = "10") int size) {

Pageable pageable = PageRequest.of(page, size);

return employeeRepository.findAll(pageable);

}

}

**b. Example URL for Pagination:**

GET /api/employees/paged?page=0&size=5

This request will return the first page of employees with 5 records per page.

**2. Sorting:**

**a. Adding Sorting Functionality to Queries**

* **Purpose:** Enhance the employee list to support sorting based on specific fields like name or email.

**EmployeeRepository Code:**

package com.example.employeemanagementsystem.repository;

import com.example.employeemanagementsystem.entity.Employee;

import org.springframework.data.domain.Page;

import org.springframework.data.domain.Pageable;

import org.springframework.data.domain.Sort;

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 fetch all employees with sorting

List<Employee> findAll(Sort sort);

// Method to fetch paginated and sorted list of employees

Page<Employee> findAll(Pageable pageable);

}

**EmployeeController Code:**

package com.example.employeemanagementsystem.controller;

import com.example.employeemanagementsystem.entity.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.PageRequest;

import org.springframework.data.domain.Pageable;

import org.springframework.data.domain.Sort;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.RestController;

import java.util.List;

@RestController

@RequestMapping("/api/employees")

public class EmployeeController {

@Autowired

private EmployeeRepository employeeRepository;

// Get sorted list of employees

@GetMapping("/sorted")

public List<Employee> getEmployeesSorted(@RequestParam(defaultValue = "name") String sortBy) {

return employeeRepository.findAll(Sort.by(sortBy));

}

// Get paginated and sorted list of employees

@GetMapping("/paged-sorted")

public Page<Employee> getEmployeesPagedAndSorted(

@RequestParam(defaultValue = "0") int page,

@RequestParam(defaultValue = "10") int size,

@RequestParam(defaultValue = "name") String sortBy) {

Pageable pageable = PageRequest.of(page, size, Sort.by(sortBy));

return employeeRepository.findAll(pageable);

}

}

**b. Example URLs for Sorting and Pagination:**

* **Sorted List:**

GET /api/employees/sorted?sortBy=email

This request will return a list of employees sorted by their email addresses.

* **Paginated and Sorted List:**

GET /api/em

ployees/paged-sorted?page=0&size=5&sortBy=name

This request will return the first page of employees, with 5 records per page, sorted by their names.