**EXERCISE 6**

*PAGINATION TO EMPLOYEE CONTROLLER*  
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.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("/employees")

public class EmployeeController {

@Autowired

private EmployeeRepository employeeRepository;

@GetMapping

public Page<Employee> getAllEmployees(

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

@RequestParam(defaultValue = "10") int size

) {

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

return employeeRepository.findAll(pageable);

}

}

*FOR HANDLING EMPTY PAGES*  
@GetMapping

public Page<Employee> getAllEmployees(

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

@RequestParam(defaultValue = "10") int size

) {

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

Page<Employee> employeePage = employeeRepository.findAll(pageable);

if (employeePage.isEmpty()) {

throw new ResourceNotFoundException("No employees found on page " + page);

}

return employeePage;

}

*SORTING FUNCTIONALITY*

# sorting with pageable

@GetMapping

public Page<Employee> getAllEmployees(

@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);

}

#sorting in asec and desc order

@GetMapping

public Page<Employee> getAllEmployees(

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

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

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

@RequestParam(defaultValue = "asc") String sortDir

) {

Sort sort = sortDir.equalsIgnoreCase("asc") ? Sort.by(sortBy).ascending() : Sort.by(sortBy).descending();

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

return employeeRepository.findAll(pageable);

}

*COMBINING PAGING AND SORTING*  
@GetMapping("/search")

public Page<Employee> searchEmployees(

@RequestParam(defaultValue = "") String name,

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

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

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

@RequestParam(defaultValue = "asc") String sortDir

) {

Sort sort = sortDir.equalsIgnoreCase("asc") ? Sort.by(sortBy).ascending() : Sort.by(sortBy).descending();

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

return employeeRepository.findByNameContaining(name, pageable);

}