EmployeeRepository.java

import org.springframework.data.domain.Page;

import org.springframework.data.domain.Pageable;

// Method to find all employees with pagination

Page<Employee> findAll(Pageable pageable);

// Method to find employees by department with pagination

Page<Employee> findByDepartmentName(String departmentName, Pageable pageable);

EmployeeController.java

import org.springframework.data.domain.Page;

import org.springframework.data.domain.PageRequest;

import org.springframework.data.domain.Pageable;

import org.springframework.data.domain.Sort;

@GetMapping("/paginated")

public ResponseEntity<Page<Employee>> getEmployeesPaginated(

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

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

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

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

return new ResponseEntity<>(employeePage, HttpStatus.OK);

}

@GetMapping("/department/paginated")

public ResponseEntity<Page<Employee>> getEmployeesByDepartmentPaginated(

@RequestParam String departmentName,

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

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

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

Page<Employee> employeePage = employeeRepository.findByDepartmentName(departmentName, pageable);

return new ResponseEntity<>(employeePage, HttpStatus.OK);

}

EmployeeRepository

Page<Employee> findAll(Pageable pageable);

Page<Employee> findByDepartmentName(String departmentName, Pageable pageable);

EmployeeController

@GetMapping("/sorted")

public ResponseEntity<Page<Employee>> getEmployeesSorted(

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

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

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

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

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

return new ResponseEntity<>(employeePage, HttpStatus.OK);

}

@GetMapping("/department/sorted")

public ResponseEntity<Page<Employee>> getEmployeesByDepartmentSorted(

@RequestParam String departmentName,

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

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

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

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

Page<Employee> employeePage = employeeRepository.findByDepartmentName(departmentName, pageable);

return new ResponseEntity<>(employeePage, HttpStatus.OK);

}

Pageable

@GetMapping("/paginated-sorted")

public ResponseEntity<Page<Employee>> getEmployeesPaginatedAndSorted(

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

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

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

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

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

return new ResponseEntity<>(employeePage, HttpStatus.OK);

}

@GetMapping("/department/paginated-sorted")

public ResponseEntity<Page<Employee>> getEmployeesByDepartmentPaginatedAndSorted(

@RequestParam String departmentName,

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

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

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

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

Page<Employee> employeePage = employeeRepository.findByDepartmentName(departmentName, pageable);

return new ResponseEntity<>(employeePage, HttpStatus.OK);

}