**WEEK 3**

**EXERCISE 4**

**Employee Management System - Implementing CRUD Operations**

**1. Basic CRUD Operations:**

**a. Creating, Reading, Updating, and Deleting Entities**

JpaRepository Methods:

* + Create: Use the save() method to create or update entities.
  + Read: Use methods like findById() or custom derived query methods to read entities.
  + Update: Use the save() method to update existing entities.
  + Delete: Use the deleteById() method to delete entities.

**b. Implementing RESTful Endpoints**

**EmployeeController**

* **Purpose:** Provides RESTful endpoints for managing Employee entities.

**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.http.ResponseEntity;

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

import java.util.List;

import java.util.Optional;

@RestController

@RequestMapping("/api/employees")

public class EmployeeController {

@Autowired

private EmployeeRepository employeeRepository;

// Create a new employee

@PostMapping

public ResponseEntity<Employee> createEmployee(@RequestBody Employee employee) {

Employee savedEmployee = employeeRepository.save(employee);

return ResponseEntity.ok(savedEmployee);

}

// Get all employees

@GetMapping

public List<Employee> getAllEmployees() {

return employeeRepository.findAll();

}

// Get employee by ID

@GetMapping("/{id}")

public ResponseEntity<Employee> getEmployeeById(@PathVariable Long id) {

Optional<Employee> employee = employeeRepository.findById(id);

return employee.map(ResponseEntity::ok).orElseGet(() -> ResponseEntity.notFound().build());

}

// Update employee

@PutMapping("/{id}")

public ResponseEntity<Employee> updateEmployee(@PathVariable Long id, @RequestBody Employee employee) {

if (employeeRepository.existsById(id)) {

employee.setId(id);

Employee updatedEmployee = employeeRepository.save(employee);

return ResponseEntity.ok(updatedEmployee);

} else {

return ResponseEntity.notFound().build();

}

}

// Delete employee

@DeleteMapping("/{id}")

public ResponseEntity<Void> deleteEmployee(@PathVariable Long id) {

if (employeeRepository.existsById(id)) {

employeeRepository.deleteById(id);

return ResponseEntity.noContent().build();

} else {

return ResponseEntity.notFound().build();

}

}

}

**DepartmentController**

* **Purpose:** Provides RESTful endpoints for managing Department entities.

**DepartmentController Code:**

package com.example.employeemanagementsystem.controller;

import com.example.employeemanagementsystem.entity.Department;

import com.example.employeemanagementsystem.repository.DepartmentRepository;

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

import org.springframework.http.ResponseEntity;

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

import java.util.List;

import java.util.Optional;

@RestController

@RequestMapping("/api/departments")

public class DepartmentController {

@Autowired

private DepartmentRepository departmentRepository;

// Create a new department

@PostMapping

public ResponseEntity<Department> createDepartment(@RequestBody Department department) {

Department savedDepartment = departmentRepository.save(department);

return ResponseEntity.ok(savedDepartment);

}

// Get all departments

@GetMapping

public List<Department> getAllDepartments() {

return departmentRepository.findAll();

}

// Get department by ID

@GetMapping("/{id}")

public ResponseEntity<Department> getDepartmentById(@PathVariable Long id) {

Optional<Department> department = departmentRepository.findById(id);

return department.map(ResponseEntity::ok).orElseGet(() -> ResponseEntity.notFound().build());

}

// Update department

@PutMapping("/{id}")

public ResponseEntity<Department> updateDepartment(@PathVariable Long id, @RequestBody Department department) {

if (departmentRepository.existsById(id)) {

department.setId(id);

Department updatedDepartment = departmentRepository.save(department);

return ResponseEntity.ok(updatedDepartment);

} else {

return ResponseEntity.notFound().build();

}

}

// Delete department

@DeleteMapping("/{id}")

public ResponseEntity<Void> deleteDepartment(@PathVariable Long id) {

if (departmentRepository.existsById(id)) {

departmentRepository.deleteById(id);

return ResponseEntity.noContent().build();

} else {

return ResponseEntity.notFound().build();

}

}

}