**Exercise 4: Employee Management System - Implementing CRUD Operations**

**Business Scenario:**

Implement CRUD operations for managing employees and departments.

**Instructions:**

1. **Basic CRUD Operations:**
   * Use **JpaRepository** methods to create, read, update, and delete employees and departments.
   * Implement RESTful endpoints for these operations using **EmployeeController** and **DepartmentController**.

**EmployeeController.java 🡪**

package com.code.employee.controllers;

import java.util.List;

import java.util.Optional;

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

import org.springframework.http.ResponseEntity;

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

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

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

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

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

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

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

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

import com.code.employee.entity.Employee;

import com.code.employee.repository.EmployeeRepository;

@RestController

    @RequestMapping("/api/employees")

    public class EmployeeController {

        @Autowired

        private EmployeeRepository employeeRepository;

        // Create a new Employee

        @PostMapping("/add")

        public Employee createEmployee(@RequestBody Employee employee) {

            return employeeRepository.save(employee);

        }

        // Get all Employees

        @GetMapping("/")

        public List<Employee> getAllEmployees() {

            return employeeRepository.findAll();

        }

        // Get a single Employee by ID

        @GetMapping("/{id}")

        public Employee getEmployeeById(@PathVariable int id) {

            Employee employee = employeeRepository.findById(id).get();

            return employee;

        }

        // Update an Employee

        @PutMapping("/edit/{id}")

        public Employee updateEmployee(@PathVariable int id, @RequestBody Employee employeeDet) {

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

            if (optionalEmployee.isPresent()) {

                Employee employee = optionalEmployee.get();

                employee.setName(employeeDet.getName());

                employee.setEmail(employeeDet.getEmail());

                employee.setDepartment(employeeDet.getDepartment());

                Employee updatedEmployee = employeeRepository.save(employee);

                return updatedEmployee;

            } else {

                return null;

            }

        }

        // Delete an Employee

        @DeleteMapping("/delete/{id}")

        public String deleteEmployee(@PathVariable int id) {

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

            if (optionalEmployee.isPresent()) {

                employeeRepository.delete(optionalEmployee.get());

                return "Deleted successfully";

            } else {

                return "Not Deleted ";

            }

        }

    }

**DepartmentController.java 🡪**

package com.code.employee.controllers;

import java.util.List;

import java.util.Optional;

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

import org.springframework.http.ResponseEntity;

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

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

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

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

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

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

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

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

import com.code.employee.entity.Department;

import com.code.employee.repository.DepartmentRepository;

@RestController

@RequestMapping("/api/departments")

public class DepartmentController {

    @Autowired

    private DepartmentRepository departmentRepository;

    // Create a new Department

    @PostMapping("/add")

    public Department createDepartment(@RequestBody Department department) {

        return departmentRepository.save(department);

    }

    // Get all Departments

    @GetMapping("/")

    public List<Department> getAllDepartments() {

        return departmentRepository.findAll();

    }

    // Get a single Department by ID

    @GetMapping("/{id}")

    public Department getDepartmentById(@PathVariable int id) {

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

        return department.get();

    }

    // Update a Department

    @PutMapping("/edit/{id}")

    public Department updateDepartment(@PathVariable int id, @RequestBody Department departmentDetails) {

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

        if (optionalDepartment.isPresent()) {

            Department department = optionalDepartment.get();

            department.setName(departmentDetails.getName());

            Department updatedDepartment = departmentRepository.save(department);

            return department;

        } else {

            return null;

        }

    }

    // Delete a Department

    @DeleteMapping("/delete/{id}")

    public String  deleteDepartment(@PathVariable int id) {

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

        if (optionalDepartment.isPresent()) {

            departmentRepository.delete(optionalDepartment.get());

            return "Department with id" +id+"deleted successfully";

        } else {

            return "Department with id" +id+" not found";

        }

    }

}