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

**EmployeeController**

package com.employee.management.system.controllers;

import com.employee.management.system.entities.Employee;

import com.employee.management.system.repositories.EmployeeRepository;

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

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

import java.util.List;

@RestController

@RequestMapping("/api/employees")

public class EmployeeController {

@Autowired

private EmployeeRepository employeeRepository;

@PostMapping

public Employee createEmployee(@RequestBody Employee employee) {

return employeeRepository.save(employee);

}

@GetMapping

public List<Employee> getAllEmployees() {

return employeeRepository.findAll();

}

@GetMapping("/{id}")

public Employee getEmployeeById(@PathVariable Long id) {

return employeeRepository.findById(id).orElse(null);

}

@PutMapping("/{id}")

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

Employee existingEmployee = employeeRepository.findById(id).orElse(null);

if (existingEmployee != null) {

existingEmployee.setName(employee.getName());

existingEmployee.setEmail(employee.getEmail());

return employeeRepository.save(existingEmployee);

}

return null;

}

@DeleteMapping("/{id}")

public void deleteEmployee(@PathVariable Long id) {

employeeRepository.deleteById(id);

}

}

**DepartmentController**

package com.employee.management.system.controllers;

import com.employee.management.system.entities.Department;

import com.employee.management.system.repositories.DepartmentRepository;

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

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

import java.util.List;

@RestController

@RequestMapping("/api/departments")

public class DepartmentController {

@Autowired

private DepartmentRepository departmentRepository;

@PostMapping

public Department createDepartment(@RequestBody Department department) {

return departmentRepository.save(department);

}

@GetMapping

public List<Department> getAllDepartments() {

return departmentRepository.findAll();

}

@GetMapping("/{id}")

public Department getDepartmentById(@PathVariable Long id) {

return departmentRepository.findById(id).orElse(null);

}

@PutMapping("/{id}")

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

Department existingDepartment = departmentRepository.findById(id).orElse(null);

if (existingDepartment != null) {

existingDepartment.setName(department.getName());

return departmentRepository.save(existingDepartment);

}

return null;

}

@DeleteMapping("/{id}")

public void deleteDepartment(@PathVariable Long id) {

departmentRepository.deleteById(id);

}

}

In the above code, we have implemented CRUD operations for managing employees and departments using Spring Data JPA and RESTful endpoints.

We have created two controllers: EmployeeController and DepartmentController, which handle requests for employee and department operations, respectively.

Each controller has methods for creating, reading, updating, and deleting employees or departments, using the corresponding repository methods.

Note that i have used @Autowired to inject the repositories into the controllers, and @RestController and @RequestMapping to define the RESTful endpoints.