**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**.

**Answer:**

1. **Basic CRUD Operations with JpaRepository:**
   * JpaRepository provides methods like save(), findById(), findAll(), deleteById(), etc., for performing CRUD operations.
2. **EmployeeController Example:**

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

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

import java.util.List;

@RestController

@RequestMapping("/employees")

public class EmployeeController {

@Autowired

private EmployeeRepository employeeRepository;

@GetMapping

public List<Employee> getAllEmployees() {

return employeeRepository.findAll();

}

@PostMapping

public Employee createEmployee(@RequestBody Employee employee) {

return employeeRepository.save(employee);

}

@GetMapping("/{id}")

public Employee getEmployeeById(@PathVariable Long id) {

return employeeRepository.findById(id).orElseThrow(() -> new ResourceNotFoundException("Employee not found"));

}

@PutMapping("/{id}")

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

Employee employee = employeeRepository.findById(id).orElseThrow(() -> new ResourceNotFoundException("Employee not found"));

employee.setName(employeeDetails.getName());

employee.setEmail(employeeDetails.getEmail());

employee.setDepartment(employeeDetails.getDepartment());

return employeeRepository.save(employee);

}

@DeleteMapping("/{id}")

public void deleteEmployee(@PathVariable Long id) {

employeeRepository.deleteById(id);

}

}

1. **DepartmentController Example:**

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

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

import java.util.List;

@RestController

@RequestMapping("/departments")

public class DepartmentController {

@Autowired

private DepartmentRepository departmentRepository;

@GetMapping

public List<Department> getAllDepartments() {

return departmentRepository.findAll();

}

@PostMapping

public Department createDepartment(@RequestBody Department department) {

return departmentRepository.save(department);

}

@GetMapping("/{id}")

public Department getDepartmentById(@PathVariable Long id) {

return departmentRepository.findById(id).orElseThrow(() -> new ResourceNotFoundException("Department not found"));

}

@PutMapping("/{id}")

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

Department department = departmentRepository.findById(id).orElseThrow(() -> new ResourceNotFoundException("Department not found"));

department.setName(departmentDetails.getName());

return departmentRepository.save(department);

}

@DeleteMapping("/{id}")

public void deleteDepartment(@PathVariable Long id) {

departmentRepository.deleteById(id);

}

}

1. **RESTful Endpoints:**
   * These controllers provide endpoints to perform CRUD operations on Employee and Department entities.
   * The endpoints follow RESTful principles, using HTTP methods (GET, POST, PUT, DELETE) for different operations.