**EXERCISE 4**

CONTROLLER

#creating an employee

@PostMapping

public Employee createEmployee(@RequestBody Employee employee) {

return employeeRepository.save(employee);

}

#reading employee

@GetMapping

public List<Employee> getAllEmployees() {

return employeeRepository.findAll();

}

@GetMapping("/{id}")

public Employee getEmployeeById(@PathVariable Long id) {

return employeeRepository.findById(id)

.orElseThrow(() -> new ResourceNotFoundException("Employee not found"));

}

#updating an employee

@PutMapping("/{id}")

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

Employee employee = employeeRepository.findById(id)

.orElseThrow(() -> new ResourceNotFoundException("Employee not found"));

employee.setName(updatedEmployee.getName());

employee.setEmail(updatedEmployee.getEmail());

employee.setDepartment(updatedEmployee.getDepartment());

return employeeRepository.save(employee);

}

#Deleting an Employee

@DeleteMapping("/{id}")

public void deleteEmployee(@PathVariable Long id) {

Employee employee = employeeRepository.findById(id)

.orElseThrow(() -> new ResourceNotFoundException("Employee not found"));

employeeRepository.deleteById(id);

}

**CRUD OPERATIONS**

#creating a dept  
@PostMapping

public Department createDepartment(@RequestBody Department department) {

return departmentRepository.save(department);

}

# reading departments

@GetMapping

public List<Department> getAllDepartments() {

return departmentRepository.findAll();

}

#updating a department

@GetMapping("/{id}")

public Department getDepartmentById(@PathVariable Long id) {

return departmentRepository.findById(id)

.orElseThrow(() -> new ResourceNotFoundException("Department not found"));

}

#Deleting a department

@DeleteMapping("/{id}")

public void deleteDepartment(@PathVariable Long id) {

Department department = departmentRepository.findById(id)

.orElseThrow(() -> new ResourceNotFoundException("Department not found"));

departmentRepository.deleteById(id);

}

*EMPLOYEE CONTOLLER*

package com.example.EmployeeManagementSystem.controller;

import com.example.EmployeeManagementSystem.model.Employee;

import com.example.EmployeeManagementSystem.repository.EmployeeRepository;

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();

}

@GetMapping("/{id}")

public Employee getEmployeeById(@PathVariable Long id) {

return employeeRepository.findById(id)

.orElseThrow(() -> new ResourceNotFoundException("Employee not found"));

}

@PostMapping

public Employee createEmployee(@RequestBody Employee employee) {

return employeeRepository.save(employee);

}

@PutMapping("/{id}")

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

Employee employee = employeeRepository.findById(id)

.orElseThrow(() -> new ResourceNotFoundException("Employee not found"));

employee.setName(updatedEmployee.getName());

employee.setEmail(updatedEmployee.getEmail());

employee.setDepartment(updatedEmployee.getDepartment());

return employeeRepository.save(employee);

}

@DeleteMapping("/{id}")

public void deleteEmployee(@PathVariable Long id) {

Employee employee = employeeRepository.findById(id)

.orElseThrow(() -> new ResourceNotFoundException("Employee not found"));

employeeRepository.deleteById(id);

}

}

*DEPARTMENT CONTROLLER*

package com.example.EmployeeManagementSystem.controller;

import com.example.EmployeeManagementSystem.model.Department;

import com.example.EmployeeManagementSystem.repository.DepartmentRepository;

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();

}

@GetMapping("/{id}")

public Department getDepartmentById(@PathVariable Long id) {

return departmentRepository.findById(id)

.orElseThrow(() -> new ResourceNotFoundException("Department not found"));

}

@PostMapping

public Department createDepartment(@RequestBody Department department) {

return departmentRepository.save(department);

}

@PutMapping("/{id}")

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

Department department = departmentRepository.findById(id)

.orElseThrow(() -> new ResourceNotFoundException("Department not found"));

department.setName(updatedDepartment.getName());

return departmentRepository.save(department);

}

@DeleteMapping("/{id}")

public void deleteDepartment(@PathVariable Long id) {

Department department = departmentRepository.findById(id)

.orElseThrow(() -> new ResourceNotFoundException("Department not found"));

departmentRepository.deleteById(id);

}

}