**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**.

**Solution**

**Employee.java**

package com.employee.employeemanagement.entity;

import jakarta.persistence.\*;

*@Entity*

*@Table*(name = "employees")

public class Employee {

*@Id*

*@GeneratedValue*(strategy = *GenerationType*.***IDENTITY***)

private Long id;

private String name;

private String email;

*@ManyToOne*

*@JoinColumn*(name = "department\_id")

private Department department;

// Constructors, Getters, Setters, etc.

public Employee() {

}

public Employee(String name, String email, Department department) {

this.name = name;

this.email = email;

this.department = department;

}

*@Override*

public String toString() {

return "Employee{" +

"id=" + id +

", name='" + name + '\'' +

", email='" + email + '\'' +

", department=" + department.getName() +

'}';

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public Department getDepartment() {

return department;

}

public void setDepartment(Department department) {

this.department = department;

}

}

**Department.java**

package com.employee.employeemanagement.entity;

import jakarta.persistence.\*;

*@Entity*

*@Table*(name = "departments")

public class Department {

*@Id*

*@GeneratedValue*(strategy = *GenerationType*.***IDENTITY***)

private Long id;

private String name;

// Constructors, Getters, Setters

public Department() {}

public Department(String name) {

this.name = name;

}

*@Override*

public String toString() {

return "Department{" +

"id=" + id +

", name='" + name + '\'' +

'}';

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

**EmployeeRepository.java**

package com.employee.employeemanagement.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import com.employee.employeemanagement.entity.Employee;

import java.util.List;

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

// Derived query methods

List<Employee> findByName(String name);

List<Employee> findByEmail(String email);

List<Employee> findByDepartmentName(String departmentName);

}

**DepartmentRepository.java**

package com.employee.employeemanagement.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import com.employee.employeemanagement.entity.Department;

import java.util.List;

public interface DepartmentRepository extends JpaRepository<Department, Long> {

// Derived query methods

List<Department> findByName(String name);

}

**EmployeeService.java**

package com.employee.employeemanagement.service;

import com.employee.employeemanagement.entity.Employee;

import com.employee.employeemanagement.repository.EmployeeRepository;

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

import org.springframework.stereotype.Service;

import java.util.List;

import java.util.Optional;

*@Service*

public class EmployeeService {

*@Autowired*

private EmployeeRepository employeeRepository;

public List<Employee> findAll() {

return employeeRepository.findAll();

}

public Optional<Employee> findById(Long id) {

return employeeRepository.findById(id);

}

public Employee save(Employee employee) {

return employeeRepository.save(employee);

}

public void deleteById(Long id) {

employeeRepository.deleteById(id);

}

}

**DepartmentService.java**

package com.employee.employeemanagement.service;

import com.employee.employeemanagement.entity.Department;

import com.employee.employeemanagement.repository.DepartmentRepository;

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

import org.springframework.stereotype.Service;

import java.util.List;

import java.util.Optional;

*@Service*

public class DepartmentService {

*@Autowired*

private DepartmentRepository departmentRepository;

public List<Department> findAll() {

return departmentRepository.findAll();

}

public Optional<Department> findById(Long id) {

return departmentRepository.findById(id);

}

public Department save(Department department) {

return departmentRepository.save(department);

}

public void deleteById(Long id) {

departmentRepository.deleteById(id);

}

}

**EmployeeController.java**

package com.employee.employeemanagement.controller;

import com.employee.employeemanagement.entity.Employee;

import com.employee.employeemanagement.service.EmployeeService;

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

import org.springframework.http.ResponseEntity;

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

import java.util.\*;

*@RestController*

*@RequestMapping*("/employees")

public class EmployeeController {

*@Autowired*

private EmployeeService employeeService;

*@GetMapping*

public List<Employee> getAllEmployees() {

return employeeService.findAll();

}

*@GetMapping*("/{id}")

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

return employeeService.findById(id)

.map(ResponseEntity::*ok*)

.orElse(ResponseEntity.*notFound*().build());

}

*@PostMapping*

public Employee createEmployee(*@RequestBody* Employee employee) {

return employeeService.save(employee);

}

*@PutMapping*("/{id}")

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

return employeeService.findById(id)

.map(employee -> {

employee.setName(employeeDetails.getName());

employee.setEmail(employeeDetails.getEmail());

employee.setDepartment(employeeDetails.getDepartment());

return ResponseEntity.*ok*(employeeService.save(employee));

})

.orElse(ResponseEntity.*notFound*().build());

}

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

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

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

if (employee.isPresent()) {

employeeService.deleteById(id);

return ResponseEntity.*noContent*().build(); // 204 No Content

} else {

return ResponseEntity.*notFound*().build(); // 404 Not Found

}

}

}

**DepartmentController.java**

package com.employee.employeemanagement.controller;

import com.employee.employeemanagement.entity.Department;

import com.employee.employeemanagement.service.DepartmentService;

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

import org.springframework.http.ResponseEntity;

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

import java.util.\*;

*@RestController*

*@RequestMapping*("/departments")

public class DepartmentController {

*@Autowired*

private DepartmentService departmentService;

*@GetMapping*

public List<Department> getAllDepartments() {

return departmentService.findAll();

}

*@GetMapping*("/{id}")

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

return departmentService.findById(id)

.map(ResponseEntity::*ok*)

.orElse(ResponseEntity.*notFound*().build());

}

*@PostMapping*

public Department createDepartment(*@RequestBody* Department department) {

return departmentService.save(department);

}

*@PutMapping*("/{id}")

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

return departmentService.findById(id)

.map(department -> {

department.setName(departmentDetails.getName());

return ResponseEntity.*ok*(departmentService.save(department));

})

.orElse(ResponseEntity.*notFound*().build());

}

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

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

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

if (department.isPresent()) {

departmentService.deleteById(id);

return ResponseEntity.*noContent*().build(); // 204 No Content

} else {

return ResponseEntity.*notFound*().build(); // 404 Not Found

}

}

}

**EmployeeManagementSystemApplication.java**

package com.employee.employeemanagement;

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import com.employee.employeemanagement.entity.Department;

import com.employee.employeemanagement.entity.Employee;

import com.employee.employeemanagement.repository.DepartmentRepository;

import com.employee.employeemanagement.repository.EmployeeRepository;

import java.util.Optional;

*@SpringBootApplication*(scanBasePackages = "com.employee.employeemanagement")

public class EmployeeManagementSystemApplication implements CommandLineRunner {

*@Autowired*

private DepartmentRepository departmentRepository;

*@Autowired*

private EmployeeRepository employeeRepository;

public static void main(String[] args) {

SpringApplication.*run*(EmployeeManagementSystemApplication.class, args);

}

*@Override*

public void run(String... args) throws Exception {

System.***out***.println();

System.***out***.println("Employee Management System is running successfully!");

// Create Department and Employee

Department cseDepartment = createDepartment("CSE");

Employee employee1 = createEmployee("Shobika", "shobika@example.com", cseDepartment);

Employee employee2 = createEmployee("Monika", "monika@example.com", cseDepartment);

// Read Employees and Departments

readEmployee(employee1.getId());

readEmployee(employee2.getId());

readDepartment(cseDepartment.getId());

// Update Employee

updateEmployee(employee1.getId(), "Shobika Updated", "shobika.updated@example.com");

// Delete Employee

deleteEmployee(employee2.getId());

// Read Employees and Departments again to see the results after update/delete

readEmployee(employee1.getId());

readDepartment(cseDepartment.getId());

}

private Department createDepartment(String name) {

Department department = new Department(name);

departmentRepository.save(department);

System.***out***.println("Created Department: " + department.getName());

return department;

}

private Employee createEmployee(String name, String email, Department department) {

Employee employee = new Employee(name, email, department);

employeeRepository.save(employee);

System.***out***.println("Created Employee: " + employee.getName() + ", Email: " + employee.getEmail() + ", Department: " + employee.getDepartment().getName());

return employee;

}

private void readEmployee(Long id) {

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

if (employee.isPresent()) {

System.***out***.println("Read Employee: " + employee.get().getName() + ", Email: " + employee.get().getEmail() + ", Department: " + employee.get().getDepartment().getName());

} else {

System.***out***.println("Employee with ID " + id + " not found.");

}

}

private void readDepartment(Long id) {

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

if (department.isPresent()) {

System.***out***.println("Read Department: " + department.get().getName());

} else {

System.***out***.println("Department with ID " + id + " not found.");

}

}

private void updateEmployee(Long id, String newName, String newEmail) {

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

if (employeeOptional.isPresent()) {

Employee employee = employeeOptional.get();

employee.setName(newName);

employee.setEmail(newEmail);

employeeRepository.save(employee);

System.***out***.println("Updated Employee: " + employee.getName() + ", Email: " + employee.getEmail());

} else {

System.***out***.println("Employee with ID " + id + " not found for update.");

}

}

private void deleteEmployee(Long id) {

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

if (employeeOptional.isPresent()) {

employeeRepository.deleteById(id);

System.***out***.println("Deleted Employee with ID: " + id);

} else {

System.***out***.println("Employee with ID " + id + " not found for deletion.");

}

}

}

