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

package com.example.employeemanagementsystem.service;

import com.example.employeemanagementsystem.entity.Employee;

import com.example.employeemanagementsystem.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;

// Create or Update Employee

public Employee saveEmployee(Employee employee) { return employeeRepository.save(employee);

}

// Read Employee by ID

public Optional<Employee> getEmployeeById(Long id) { return employeeRepository.findById(id);

}

// Read All Employees

public List<Employee> getAllEmployees() { return employeeRepository.findAll();

}

// Delete Employee by ID

public void deleteEmployee(Long id) { employeeRepository.deleteById(id);

}

}

DepartmentService

package com.example.employeemanagementsystem.service;

import com.example.employeemanagementsystem.entity.Department;

import com.example.employeemanagementsystem.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;

// Create or Update Department

public Department saveDepartment(Department department) { return departmentRepository.save(department);

}

// Read Department by ID

public Optional<Department> getDepartmentById(Long id) {

return departmentRepository.findById(id);

}

// Read All Departments

public List<Department> getAllDepartments() {

return departmentRepository.findAll();

}

// Delete Department by ID

public void deleteDepartment(Long id) {

departmentRepository.deleteById(id);

}

}

Implementing RESTful Endpoints in Controllers EmployeeController

package com.example.employeemanagementsystem.controller;

import com.example.employeemanagementsystem.entity.Employee;

import com.example.employeemanagementsystem.service.EmployeeService;

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

import org.springframework.http.ResponseEntity;

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

import java.util.List;

import java.util.Optional;

@RestController

@RequestMapping("/api/employees") public class EmployeeController {

@Autowired

private EmployeeService employeeService;

// Create a new Employee @PostMapping

public Employee createEmployee(@RequestBody Employee employee) {

return employeeService.saveEmployee(employee);

}

// Get an Employee by ID @GetMapping("/{id}")

public ResponseEntity<Employee> getEmployeeById(@PathVariable Long id) { Optional<Employee> employee = employeeService.getEmployeeById(id);

return employee.map(ResponseEntity::ok).orElseGet(() -> ResponseEntity.notFound().build());

}

// Get all Employees @GetMapping

public List<Employee> getAllEmployees() {

return employeeService.getAllEmployees();

}

// Update an Employee @PutMapping("/{id}")

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

Optional<Employee> employee = employeeService.getEmployeeById(id); if (employee.isPresent()) {

Employee existingEmployee = employee.get();

existingEmployee.setName(employeeDetails.getName()); existingEmployee.setEmail(employeeDetails.getEmail());

existingEmployee.setDepartment(employeeDetails.getDepartment());

Employee updatedEmployee = employeeService.saveEmployee(existingEmployee);

return ResponseEntity.ok(updatedEmployee);

} else {

return ResponseEntity.notFound().build();

}

}

// Delete an Employee @DeleteMapping("/{id}")

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

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

if (employee.isPresent()) {

employeeService.deleteEmployee(id);

return ResponseEntity.noContent().build();

} else {

return ResponseEntity.notFound().build();

}

}

}

DepartmentController

package com.example.employeemanagementsystem.controller;

import com.example.employeemanagementsystem.entity.Department;

import com.example.employeemanagementsystem.service.DepartmentService;

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

import org.springframework.http.ResponseEntity;

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

import java.util.List;

import java.util.Optional;

@RestController

@RequestMapping("/api/departments") public class DepartmentController {

@Autowired

private DepartmentService departmentService;

// Create a new Department @PostMapping

public Department createDepartment(@RequestBody Department department) {

return departmentService.saveDepartment(department);

}

// Get a Department by ID @GetMapping("/{id}")

public ResponseEntity<Department> getDepartmentById(@PathVariable Long id) { Optional<Department> department = departmentService.getDepartmentById(id);

return department.map(ResponseEntity::ok).orElseGet(() -> ResponseEntity.notFound().build());

}

// Get all Departments @GetMapping

public List<Department> getAllDepartments() {

return departmentService.getAllDepartments();

}

// Update a Department @PutMapping("/{id}")

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

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

if (department.isPresent()) {

Department existingDepartment = department.get();

existingDepartment.setName(departmentDetails.getName());

Department updatedDepartment = departmentService.saveDepartment(existingDepartment);

return ResponseEntity.ok(updatedDepartment);

} else {

return ResponseEntity.notFound().build();

}

}

// Delete a Department @DeleteMapping("/{id}")

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

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

if (department.isPresent()) {

departmentService.deleteDepartment(id);

return ResponseEntity.noContent().build();

} else {

return ResponseEntity.notFound().build();

}

}

}

* **RESTful Endpoints**:
  + @PostMapping: To create a new entity (Employee or Department).
  + @GetMapping: To retrieve entities by ID or list all entities.
  + @PutMapping: To update an existing entity.
  + @DeleteMapping: To delete an entity by ID.
* **ResponseEntity**: Provides a standardized way to return HTTP responses, including status codes (e.g., 200 OK, 404 Not Found, 204 No Content).

**Final Steps**

1. **Test the API Endpoints**:
   * Use tools like Postman or curl to send HTTP requests to the API endpoints.
   * Ensure all CRUD operations work as expected.
2. **Expand Functionality**:
   * Implement additional business logic as needed.
   * Add validation, exception handling, and more complex queries or services.

Your Employee Management System now supports full CRUD operations via RESTful endpoints for both employees and departments.