**Exercise 2: Employee Management System - Creating Entities**

**Business Scenario:**

Define JPA entities for Employee and Department with appropriate relationships.

**Instructions:**

1. **Creating JPA Entities:**
   * Define **Employee** entity with fields: **id, name, email, department**.
   * Define **Department** entity with fields: **id, name**.
2. **Mapping Entities to Database Tables:**
   * Use annotations like **@Entity, @Table, @Id, @GeneratedValue**, etc.

Define one-to-many relationship between **Department** and **Employee**

**Solution**

**Employee.java**

package com.employee.employeemanagement.entity;

import javax.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;

public Employee() {

}

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

this.name = name;

this.email = email;

this.department = department;

}

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 javax.persistence.\*;

import java.util.List;

*@*Entity

*@*Table(name = "departments")

public class Department {

*@*Id

*@*GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

*@*OneToMany(mappedBy = "department", cascade = CascadeType.ALL, orphanRemoval = true)

private List<Employee> employees;

// Constructors, Getters, Setters, etc.

public Department() {

}

public Department(String name) {

this.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;

}

public List<Employee> getEmployees() {

return employees;

}

public void setEmployees(List<Employee> employees) {

this.employees = employees;

}

}

**EmployeeManagementSystemApplication.java**

package com.employee.employeemanagement;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import com.employee.employeemanagement.entity.Department;

import com.employee.employeemanagement.entity.Employee;

*@SpringBootApplication*

public class EmployeeManagementSystemApplication implements CommandLineRunner {

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 a Department instance

Department cseDepartment = new Department("CSE");

// Create Employee instances

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

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

// Print employee details to the console

System.***out***.println("Created Employee 1: " + employee1.getName() + ", Email: " + employee1.getEmail() + ", Department: " + employee1.getDepartment().getName());

System.***out***.println("Created Employee 2: " + employee2.getName() + ", Email: " + employee2.getEmail() + ", Department: " + employee2.getDepartment().getName());

}

}

**pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.3.2</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.employee.employeemanagement</groupId>

<artifactId>EmployeeManagementSystem</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>EmployeeManagementSystem</name>

<description>Employee Management System using Spring Boot</description>

<url/>

<licenses>

<license/>

</licenses>

<developers>

<developer/>

</developers>

<scm>

<connection/>

<developerConnection/>

<tag/>

<url/>

</scm>

<properties>

<java.version>17</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

<dependency>

<groupId>javax.persistence</groupId>

<artifactId>javax.persistence-api</artifactId>

<version>2.2</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

<configuration>

<excludes>

<exclude>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

</exclude>

</excludes>

</configuration>

</plugin>

</plugins>

</build>

</project>

