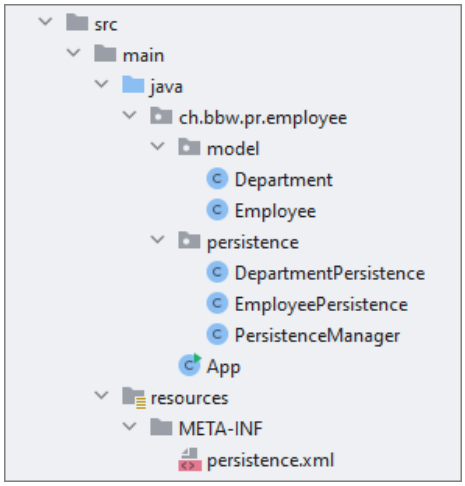
Umsetzung JPA, Hibernate mit Entity Manger, Beziehung Musterlösung

Projektübersicht:



Maven Dependencies:

```
<!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
<dependency>
  <groupId>org.hibernate</groupId>
  <artifactId>hibernate-core</artifactId>
  <version>5.5.7.Final</version>
</dependency>
<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>8.0.26</version>
</dependency>
</dependencies>
```

Konfiguration

persistence.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1"
    xmlns="http://xmlns.jcp.org/xml/ns/persistence"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence/persistence
    <persistence-unit name="MyPersistanceUnit" transaction-type="RESOURCE_LOCAL">
        <class>ch.bbw.pr.employee.model.Employee</class>
        <class>ch.bbw.pr.employee.model.Department</class>
        <properties>
            <property name="hibernate.dialect" value="org.hibernate.dialect.MySQL5Dialect"/>
            <property name="hibernate.hbm2ddl.auto" value="update" />
            <property name="javax.persistence.jdbc.driver" value="com.mysql.cj.jdbc.Driver" />
            <property name="javax.persistence.jdbc.url" value="jdbc:mysql://127.0.0.1:3306/employeedepartment" />
            <property name="javax.persistence.jdbc.user" value="root" />
            <property name="javax.persistence.jdbc.password" value="1234" />
        </properties>
    </persistence-unit>
</persistence>
```

Auszug aus Application Class

App.java

```java
package ch.bbw.pr.employee;

import ch.bbw.pr.employee.model.Department;
import ch.bbw.pr.employee.model.Employee;
import ch.bbw.pr.employee.persistence.DepartmentPersistence;
import ch.bbw.pr.employee.persistence.EmployeePersistence;
import ch.bbw.pr.employee.persistence.PersistenceManager;

import java.util.List;

/**
 * Applicatioin class
 * @author Peter Rutschmann
 * @version 17.09.2021
 */
public class App
{
    public static void main( String[] args )
    {
        System.out.println( "Employee with Hibernate Entity Manager and Relationships" );
        EmployeePersistence persistence = PersistenceManager.getEmployeePersistence();

        System.out.println("List all employees: \n" + persistence.getAllEmployees());
        System.out.println();

        System.out.println("Add new Employee");
        Employee emp = new Employee();
        emp.setFirstname("Lars");
        emp.setLastname("Lustig");
        persistence.createEmployee(emp);
        System.out.println("List all employees: \n" + persistence.getAllEmployees());
        System.out.println();
```

...

```java
        //-----------------------------------------
        System.out.println("Add new Employee with existing Department");
        dpList = departmentPersistence.getAllDepartments();
        if ((dpList != null) && (dpList.size()>1)) {
            Department dep2 = departmentPersistence.readDepartment(dpList.get(dpList.size()-1).getId());
            System.out.println(dep2);

            Employee emp2 = new Employee();
            emp2.setFirstname("Niklaus");
            emp2.setLastname("Waldmann");
            emp2.setDepartment(dep2);
            persistence.createEmployee(emp2);
        }
        System.out.println("List all employees: \n" + persistence.getAllEmployees());
        System.out.println();
```

Employee.java

```java
package ch.bbw.pr.employee.model;

import javax.persistence.*;

/**
 * Employee
 *
 * @author Peter Rutschmann
 * @version 17.09.2021
 */
@Entity
@Table(name = "employee")
@NamedQuery(name = "Employee.findAll", query = "FROM Employee")
public class Employee {
    @Id
    @Column(name = "id", unique = true)
    @GeneratedValue(strategy=GenerationType.AUTO)
    private int id;

    @Column(name = "firstname")
    private String firstname;

    @Column(name = "lastname")
    private String lastname;

    //@Column(name = "departmentidfs")
    //private Integer departmentidfs;

    @ManyToOne
    @JoinColumn(name = "departmentidfs")
    private Department department;

    public Employee() {
    }

    public int getId() { return id; }

    public void setId(int id) { this.id = id; }

    public String getFirstname() { return firstname; }

    public void setFirstname(String firstname) { this.firstname = firstname; }

    public String getLastname() { return lastname; }

    public void setLastname(String lastname) { this.lastname = lastname; }

   /* public Integer getDepartmentidfs() {
        return departmentidfs;
    }

    public void setDepartmentidfs(Integer departmentidfs) {
        this.departmentidfs = departmentidfs;
    }

    */

    public Department getDepartment() { return department; }

    public void setDepartment(Department department) { this.department = department; }

    @Override
    public String toString() {
        return "Employee{" +
                "id=" + id +
                ", firstname='" + firstname + '\'' +
                ", lastname='" + lastname + '\'' +
                //", departmentidfs=" + departmentidfs +
                ", department=" + department +
                '}';
    }
}
```

Department.java

```java
package ch.bbw.pr.employee.model;

import javax.persistence.*;

/**
 * Department
 *
 * @author Peter Rutschmann
 * @version 17.09.2021
 */
@Entity
@Table(name = "department")
@NamedQuery(name = "Department.findAll", query = "FROM Department")
public class Department {
    @Id
    @Column(name = "id", unique = true)
    private int id;

    @Column(name = "description")
    private String description;

    public Department() {
    }

    public int getId() { return id; }

    public void setId(int id) { this.id = id; }

    public String getDescription() { return description; }

    public void setDescription(String description) { this.description = description; }

    @Override
    public String toString() {
        return "Department{" +
                "id=" + id +
                ", description='" + description + '\'' +
                '}';
    }
}
```

Verwaltet die Persistence Objekte.
Es soll jeweils nur eines davon geben… und so die Zugriffe auf die Datenbank kanalisieren.

PersistenceManager.java

```
import ...

/**
 * PersistenceManager
 *
 * @author Peter Rutschmann
 * @version 17.09.2021
 */
public class PersistenceManager {
   private static EntityManagerFactory emfactory;
   private static EntityManager entitymanager;
   private static EmployeePersistence employeePersistence;
   private static DepartmentPersistence deploymentPersistence;

   public static void close(){
      entitymanager.close();
      emfactory.close();
   }

   public static EmployeePersistence getEmployeePersistence() {
      if (employeePersistence == null){
         if (entitymanager == null){
            emfactory = Persistence.createEntityManagerFactory( persistenceUnitName: "MyPersistanceUnit");
            entitymanager = emfactory.createEntityManager();
         }
         employeePersistence = new EmployeePersistence(entitymanager);
      }
      return employeePersistence;
   }

   public static DepartmentPersistence getDeploymentPersistence() {
      if (deploymentPersistence == null){
         if (entitymanager == null){
            emfactory = Persistence.createEntityManagerFactory( persistenceUnitName: "MyPersistanceUnit");
            entitymanager = emfactory.createEntityManager();
         }
         deploymentPersistence = new DepartmentPersistence(entitymanager);
      }
      return deploymentPersistence;
   }
}
```

EmployeePersistence.java

```java
package ch.bbw.pr.employee.persistence;

import ch.bbw.pr.employee.model.Employee;

import javax.persistence.EntityManager;
import java.util.List;

/**
 * EmployeePersistence
 *
 * @author Peter Rutschmann
 * @version 17.09.2021
 */
public class EmployeePersistence {
   private static EntityManager entitymanager;

   public EmployeePersistence(EntityManager entitymanager) { this.entitymanager = entitymanager; }

   public List<Employee> getAllEmployees(){
      List employees = null;

      try {
         entitymanager.getTransaction().begin();
         //employees = entitymanager.createQuery("from Employee").getResultList();
         employees = entitymanager.createNamedQuery("Employee.findAll").getResultList();
         entitymanager.getTransaction().commit();
      } catch (Exception e) {
         e.printStackTrace();
         entitymanager.getTransaction().rollback();
      }

      return employees;
   }

   public void createEmployee(Employee employee) {
      try {
         entitymanager.getTransaction().begin();
         entitymanager.persist( employee );
         entitymanager.getTransaction().commit();
      } catch (Exception e) {
         e.printStackTrace();
         entitymanager.getTransaction().rollback();
      }
   }

   public Employee readEmployee(int id) {
      Employee dbEmployee=null;
      try {
         entitymanager.getTransaction().begin();
         dbEmployee=entitymanager.find( Employee.class, id);
         entitymanager.getTransaction().commit();
      } catch (Exception e) {
         e.printStackTrace();
         entitymanager.getTransaction().rollback();
      }
      return dbEmployee;
   }

   public void updateEmployee(Employee employee) {
      try {
         entitymanager.getTransaction().begin();
         Employee dbEmployee=entitymanager.find( Employee.class, employee.getId() );
         if (dbEmployee != null) {
            entitymanager.merge(employee);
         }
         entitymanager.getTransaction().commit();
      } catch (Exception e) {
         e.printStackTrace();
         entitymanager.getTransaction().rollback();
      }
   }
```

```
74
75      public void deleteEmployee(int id) {
76         try {
77            entitymanager.getTransaction().begin();
78            Employee dbEmployee=entitymanager.find( Employee.class, id);
79            if (dbEmployee != null) {
80               entitymanager.remove( dbEmployee );
81            }
82            entitymanager.getTransaction().commit();
83         } catch (Exception e) {
84            e.printStackTrace();
85            entitymanager.getTransaction().rollback();
86         }
87      }
88   }
```

DepartmentPersistence.java

```
1   package ch.bbw.pr.employee.persistence;
2
3   import ch.bbw.pr.employee.model.Department;
4
5   import javax.persistence.EntityManager;
6   import javax.persistence.EntityManagerFactory;
7   import java.util.List;
8
9   /**
10   * DepartmentPersistence
11   *
12   * @author Peter Rutschmann
13   * @version 17.09.2021
14   */
15  public class DepartmentPersistence {
16     private static EntityManager entitymanager;
17
18     public DepartmentPersistence(EntityManager entitymanager) { this.entitymanager = entitymanager; }
21
22     public List<Department> getAllDepartments(){...}
35
36     public void createDepartment(Department department) {...}
46
47     public Department readDepartment(int id) {...}
59
60     public void updateDepartment(Department department) {...}
73
74     public void deleteDepartment(int id) {...}
87  }
```

Der Inhalt der Methoden ist (fast) gleich wie beim EmployeePersistence