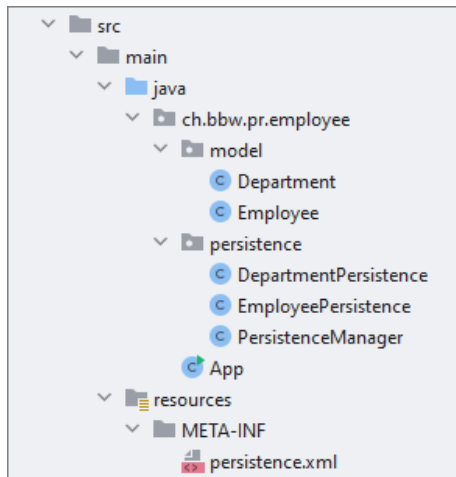


Umsetzung JPA, Hibernate mit Entity Manger, Beziehung

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

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <persistence version="2.1"
3   xmlns="http://xmlns.jcp.org/xml/ns/persistence"
4   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5   xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd"
6   <persistence-unit name="MyPersistenceUnit" transaction-type="RESOURCE_LOCAL">
7     <class>ch.bbw.pr.employee.model.Employee</class>
8     <class>ch.bbw.pr.employee.model.Department</class>
9     <properties>
10      <property name="hibernate.dialect" value="org.hibernate.dialect.MySQL5Dialect"/>
11      <property name="hibernate.hbm2ddl.auto" value="update" />
12      <property name="javax.persistence.jdbc.driver" value="com.mysql.cj.jdbc.Driver" />
13      <property name="javax.persistence.jdbc.url" value="jdbc:mysql://127.0.0.1:3306/employee_department" />
14      <property name="javax.persistence.jdbc.user" value="root" />
15      <property name="javax.persistence.jdbc.password" value="1234" />
16    </properties>
17  </persistence-unit>
18 </persistence>
  
```

Auszug aus Application Class

```

App.java x
1  package ch bbw.pr.employee;
2
3  import ch bbw.pr.employee.model.Department;
4  import ch bbw.pr.employee.model.Employee;
5  import ch bbw.pr.employee.persistence.DepartmentPersistence;
6  import ch bbw.pr.employee.persistence.EmployeePersistence;
7  import ch bbw.pr.employee.persistence.PersistenceManager;
8
9  import java.util.List;
10
11  /**
12   * Application class
13   * @author Peter Rutschmann
14   * @version 17.09.2021
15   */
16  public class App
17  {
18      public static void main( String[] args )
19      {
20          System.out.println( "Employee with Hibernate Entity Manager and Relationships" );
21          EmployeePersistence persistence = PersistenceManager.getEmployeePersistence();
22
23          System.out.println("List all employees: \n" + persistence.getAllEmployees());
24          System.out.println();
25
26          System.out.println("Add new Employee");
27          Employee emp = new Employee();
28          emp.setFirstname("Lars");
29          emp.setLastname("Lustig");
30          persistence.createEmployee(emp);
31          System.out.println("List all employees: \n" + persistence.getAllEmployees());
32          System.out.println();

```

...

```

105  //-----
106  System.out.println("Add new Employee with existing Department");
107  dpList = departmentPersistence.getAllDepartments();
108  if ((dpList != null) && (dpList.size()>1)) {
109      Department dep2 = departmentPersistence.readDepartment(dpList.get(dpList.size()-1).getId());
110      System.out.println(dep2);
111
112      Employee emp2 = new Employee();
113      emp2.setFirstname("Niklaus");
114      emp2.setLastname("Waldmann");
115      emp2.setDepartment(dep2);
116      persistence.createEmployee(emp2);
117  }
118  System.out.println("List all employees: \n" + persistence.getAllEmployees());
119  System.out.println();

```

```

1 package ch.bbw.pr.employee.model;
2
3 import javax.persistence.*;
4
5 /**
6  * Employee
7  *
8  * @author Peter Rutschmann
9  * @version 17.09.2021
10 */
11 @Entity
12 @Table(name = "employee")
13 @NamedQuery(name = "Employee.findAll", query = "FROM Employee")
14 public class Employee {
15     @Id
16     @Column(name = "id", unique = true)
17     @GeneratedValue(strategy=GenerationType.AUTO)
18     private int id;
19
20     @Column(name = "firstname")
21     private String firstname;
22
23     @Column(name = "lastname")
24     private String lastname;
25
26     // @Column(name = "departmentidfs")
27     // private Integer departmentidfs;
28
29     @ManyToOne
30     @JoinColumn(name = "departmentidfs")
31     private Department department;
32
33     public Employee() {
34     }
35
36     public int getId() { return id; }
37
38     public void setId(int id) { this.id = id; }
39
40     public String getFirstname() { return firstname; }
41
42     public void setFirstname(String firstname) { this.firstname = firstname; }
43
44     public String getLastname() { return lastname; }
45
46     public void setLastname(String lastname) { this.lastname = lastname; }
47
48     /* public Integer getDepartmentidfs() {
49         return departmentidfs;
50     }
51
52     public void setDepartmentidfs(Integer departmentidfs) {
53         this.departmentidfs = departmentidfs;
54     }
55
56     */
57
58     public Department getDepartment() { return department; }
59
60     public void setDepartment(Department department) { this.department = department; }
61
62     @Override
63     public String toString() {
64         return "Employee{" +
65             "id=" + id +
66             ", firstname='" + firstname + '\'' +
67             ", lastname='" + lastname + '\'' +
68             // ", departmentidfs=" + departmentidfs +
69             ", department=" + department +
70             '}';
71     }
72 }

```

```
1 package ch bbw.pr.employee.model;
2
3 import javax.persistence.*;
4
5 /**
6  * Department
7  *
8  * @author Peter Rutschmann
9  * @version 17.09.2021
10 */
11 @Entity
12 @Table(name = "department")
13 @NamedQuery(name = "Department.findAll", query = "FROM Department")
14 public class Department {
15     @Id
16     @Column(name = "id", unique = true)
17     private int id;
18
19     @Column(name = "description")
20     private String description;
21
22     public Department() {
23     }
24
25     public int getId() { return id; }
26
27
28     public void setId(int id) { this.id = id; }
29
30
31     public String getDescription() { return description; }
32
33
34     public void setDescription(String description) { this.description = description; }
35
36
37     @Override
38     public String toString() {
39         return "Department{" +
40             "id=" + id +
41             ", description='" + description + '\'' +
42             '}';
43     }
44 }
45
46
47
48 }
```

Verwaltet die Persistence Objekte.

Es soll jeweils nur eines davon geben... und so die Zugriffe auf die Datenbank kanalisieren.

```
PersistenceManager.java
3  import ...
4
5  /**
6   * PersistenceManager
7   *
8   * @author Peter Rutschmann
9   * @version 17.09.2021
10  */
11
12  public class PersistenceManager {
13      private static EntityManagerFactory emfactory;
14      private static EntityManager entitymanager;
15      private static EmployeePersistence employeePersistence;
16      private static DepartmentPersistence deploymentPersistence;
17
18
19      public static void close(){
20          entitymanager.close();
21          emfactory.close();
22      }
23
24      public static EmployeePersistence getEmployeePersistence() {
25          if (employeePersistence == null){
26              if (entitymanager == null){
27                  emfactory = Persistence.createEntityManagerFactory( persistenceUnitName: "MyPersistenceUnit");
28                  entitymanager = emfactory.createEntityManager();
29              }
30              employeePersistence = new EmployeePersistence(entitymanager);
31          }
32          return employeePersistence;
33      }
34
35      public static DepartmentPersistence getDeploymentPersistence() {
36          if (deploymentPersistence == null){
37              if (entitymanager == null){
38                  emfactory = Persistence.createEntityManagerFactory( persistenceUnitName: "MyPersistenceUnit");
39                  entitymanager = emfactory.createEntityManager();
40              }
41              deploymentPersistence = new DepartmentPersistence(entitymanager);
42          }
43          return deploymentPersistence;
44      }
45  }
```

```
EmployeePersistence.java x
1 package ch.bbw.pr.employee.persistence;
2
3 import ch.bbw.pr.employee.model.Employee;
4
5 import javax.persistence.EntityManager;
6 import java.util.List;
7
8 /**
9  * EmployeePersistence
10  *
11  * @author Peter Rutschmann
12  * @version 17.09.2021
13  */
14 public class EmployeePersistence {
15     private static EntityManager entitymanager;
16
17     public EmployeePersistence(EntityManager entitymanager) { this.entitymanager = entitymanager; }
18
19     public List<Employee> getAllEmployees(){
20         List employees = null;
21
22         try {
23             entitymanager.getTransaction().begin();
24             //employees = entitymanager.createQuery("from Employee").getResultList();
25             employees = entitymanager.createNamedQuery("Employee.findAll").getResultList();
26             entitymanager.getTransaction().commit();
27         } catch (Exception e) {
28             e.printStackTrace();
29             entitymanager.getTransaction().rollback();
30         }
31
32         return employees;
33     }
34
35     public void createEmployee(Employee employee) {
36         try {
37             entitymanager.getTransaction().begin();
38             entitymanager.persist( employee );
39             entitymanager.getTransaction().commit();
40         } catch (Exception e) {
41             e.printStackTrace();
42             entitymanager.getTransaction().rollback();
43         }
44     }
45
46     public Employee readEmployee(int id) {
47         Employee dbEmployee=null;
48         try {
49             entitymanager.getTransaction().begin();
50             dbEmployee=entitymanager.find( Employee.class, id);
51             entitymanager.getTransaction().commit();
52         } catch (Exception e) {
53             e.printStackTrace();
54             entitymanager.getTransaction().rollback();
55         }
56         return dbEmployee;
57     }
58
59     public void updateEmployee(Employee employee) {
60         try {
61             entitymanager.getTransaction().begin();
62             Employee dbEmployee=entitymanager.find( Employee.class, employee.getId() );
63             if (dbEmployee != null) {
64                 entitymanager.merge(employee);
65             }
66             entitymanager.getTransaction().commit();
67         } catch (Exception e) {
68             e.printStackTrace();
69             entitymanager.getTransaction().rollback();
70         }
71     }
72
73 }
```

```

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 }

```

```

C DepartmentPersistence.java x
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; }
19
20
21
22         public List<Department> getAllDepartments() {...}
23
24
25
26
27
28
29
30
31
32
33
34
35
36         public void createDepartment(Department department) {...}
37
38
39
40
41
42
43
44
45
46
47         public Department readDepartment(int id) {...}
48
49
50
51
52
53
54
55
56
57
58
59         public void updateDepartment(Department department) {...}
60
61
62
63
64
65
66
67
68
69
70
71
72
73         public void deleteDepartment(int id) {...}
74
75
76
77
78
79
80
81
82
83
84
85
86
87     }

```

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