

Umsetzung JPA, Hibernate mit Entity Manger



Einleitung:

Sie setzen dieses Beispiel um:

- ⇒ Java Projekt mit Maven, Verwendung von JPA, Hibernate EntityManager.
- ⇒ In dieser Variante übernimmt der EntityManager einen Teil der Arbeit.

Ziele:

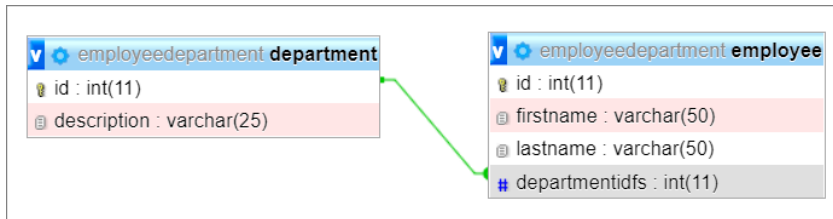
- ⇒ Sie können Entitäten aus einer Datenbank über das Persistenz-Framework und das Interface JPA mit Objekten der Applikation verwalten.
- ⇒ Sie setzen den EntityManager ein.

Inhaltsverzeichnis

Einleitung:	1
Ziele:	1
Inhaltsverzeichnis	1
Aufgabe "Datenbank bereitstellen"	2
Aufgabe "Maven Projekt erstellen "	2
Aufgabe "Projekt implementieren"	2
Variante mit @NamedQuery	6

Aufgabe "Datenbank bereitstellen"

- Erstellen Sie auf Ihrem Datenbankserver die neue Datenbank **employeedepartment**.
- Importieren Sie die Tabellen und Daten. *employeedepartment.sql*



Für die Beziehung ist voreingestellt: department mc – c employee

Aufgabe "Maven Projekt erstellen "

Erstellen Sie ein neues Maven Projekt.

Hilfe? Siehe → 01_3_MavenSchrittUmSchrittIntelliJ.pdf

Ergänzen Sie die **Dependencies** für *hibernate* und *mysql*.

Die aktuellen findet man im *mvnrepository*. Hier die von mir verwendeten Versionen.

<https://mvnrepository.com/artifact/org.hibernate/hibernate-core>

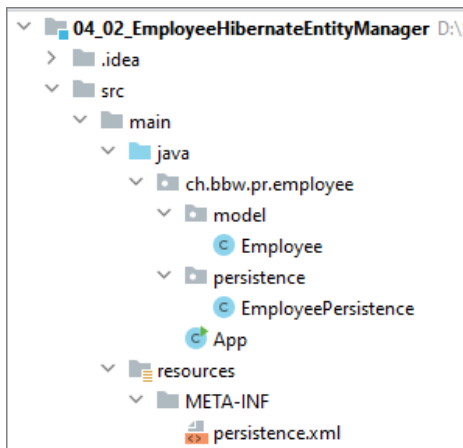
```
<!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
<dependency>
  <groupId>org.hibernate</groupId>
  <artifactId>hibernate-core</artifactId>
  <version>5.5.7.Final</version>
</dependency>
```

<https://mvnrepository.com/artifact/mysql/mysql-connector-java/8.0.26>

```
<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>8.0.26</version>
</dependency>
```

Aufgabe "Projekt implementieren"

Es folgen nun Erweiterungen so dass diese Struktur entsteht.



Employee ist die Datenklasse

EmployeePersistence macht den Datenaustausch mit der Datenbank

App ist die Applikationsklasse

persistence.xml ist die Konfigurationsdatei für den Datenbankzugriff.

Beachten Sie den dazugehörigen Ordner META-INF

Employee.java

Implementieren Sie die Klasse Employee mit den Attributen

- id:int
- firstname:String
- lastname:String
- departmentidfs:Integer
- Constructor, Getter und Setter und der toString Methode

```
Employee.java x
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 public class Employee {
14     @Id
15     @Column(name = "id", unique = true)
16     private int id;
17
18     @Column(name = "firstname")
19     private String firstname;
20
21     @Column(name = "lastname")
22     private String lastname;
23
24     @Column(name = "departmentidfs")
25     private Integer departmentidfs;
```

Fügen Sie diese Annotierungen für JPA hinzu.
Sie beginnen alle mit @

@Entity

@Table(name = "employee")

public class Employee {

@Id

@Column(name = "id", unique = true)

private int id;

@Column(name = "firstname")

private String firstname;

@Column(name = "lastname")

private String lastname;

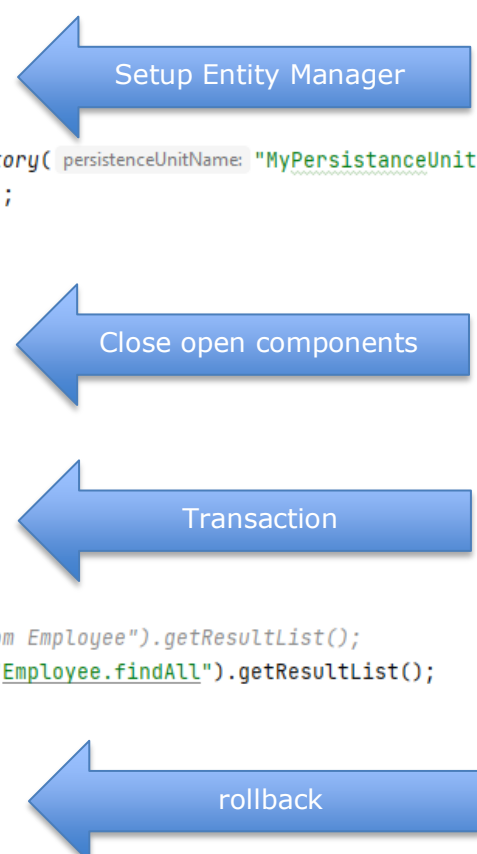
@Column(name = "departmentidfs")

private Integer departmentidfs;

EmployeePersistence.java

Implementieren Sie die Klasse *EmployeePersistence.java*

```
16 public class EmployeePersistence {
17     private static EntityManagerFactory emfactory;
18     private static EntityManager entitymanager;
19
20     public EmployeePersistence(){
21         //Setup EntityManager
22         emfactory = Persistence.createEntityManagerFactory( persistenceUnitName: "MyPersistenceUnit");
23         entitymanager = emfactory.createEntityManager();
24     }
25
26     public void close(){
27         entitymanager.close();
28         emfactory.close();
29     }
30
31     public List<Employee> getAllEmployees(){
32         List employees = null;
33
34         try {
35             entitymanager.getTransaction().begin();
36             //employees = entitymanager.createQuery("from Employee").getResultList();
37             employees = entitymanager.createNamedQuery("Employee.findAll").getResultList();
38             entitymanager.getTransaction().commit();
39         } catch (Exception e) {
40             e.printStackTrace();
41             entitymanager.getTransaction().rollback();
42         }
43
44         return employees;
45     }
46 }
```



Die Klasse behandelt den Zugriff auf die Datenbank mit Hilfe des EntityManager.

Dieser verwaltet im Hintergrund die Session und die Transaction mit der DB.

App.java

Implementieren Sie die App Klasse

- Instanzieren eines Objektes von *EmployeePersistence*
- Aufruf der Methode *getAllEmployees* und Ausgabe auf Console.
- Aufruf der Methode *close*.

```

App.java
1 package ch.bbw.pr.employee;
2
3 import ch.bbw.pr.employee.persistence.EmployeePersistence;
4
5 /**
6  * Application class
7  * @author Peter Rutschmann
8  * @version 17.09.2021
9  */
10 public class App
11 {
12     public static void main( String[] args )
13     {
14         System.out.println( "Employee with Hibernate Entity Manager" );
15         EmployeePersistence persistence = new EmployeePersistence();
16         System.out.println("List all employees: " + persistence.getAllEmployees());
17         persistence.close();
18     }
19 }

```

persistence.xml

Die Konfiguration über die EntityManagerFactory basiert auf dieser Konfigurationsdatei im Ordner METH-INF

Achten Sie bei der Implementierung auf die Unterstützung Ihrer IDE!

Dank dem xsd weiss die IDE, welche Token möglich sind.

Übernehmen Sie die ersten Zeilen:

```

<?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_2_1.xsd">

```

und ergänzen Sie die Datei.

```

persistence.xml
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         <properties>
9             <property name="hibernate.dialect" value="org.hibernate.dialect.MySQL5Dialect"/>
10            <property name="hibernate.hbm2ddl.auto" value="update" />
11            <property name="javax.persistence.jdbc.driver" value="com.mysql.cj.jdbc.Driver" />
12            <property name="javax.persistence.jdbc.url" value="jdbc:mysql://127.0.0.1:3306/employeeedepartment" />
13            <property name="javax.persistence.jdbc.user" value="root" />
14            <property name="javax.persistence.jdbc.password" value="1234" />
15        </properties>
16    </persistence-unit>
17 </persistence>

```

Variante mit @NamedQuery

Eine Query kann bei der Entity definiert und über einen Namen referenziert werden.

Das hat den Vorteil, dass die Query nah bei der Entität ist, zu der sie gehört.

Aufruf der NamedQuery

```
31 public List<Employee> getAllEmployees(){
32     List employees = null;
33
34     entityManager.getTransaction().begin();
35     //employees = entityManager.createQuery("from Employee").getResultList();
36     employees = entityManager.createNamedQuery("Employee.findAll").getResultList();
37     entityManager.getTransaction().commit();
38
39     return employees;
40 }
```

Die Query deklariert bei der Klasse Employee

```
11 @Entity
12 @Table(name = "employee")
13 @NamedQuery(name = "Employee.findAll", query = "FROM Employee")
14 public class Employee {
```

Für die Schnellen... probieren Sie es aus.