Umsetzung JPA, Hibernate Session und Transaktion

Einleitung:

Es gibt ganz viele Tutorials im Internet. Doch nicht immer ist es so einfach diese zum Laufen zu bringen.

Hier eine Auswahl:

https://examples.javacodegeeks.com/enterprise-java/jpa/jpa-entitymanager-example/

https://examples.javacodegeeks.com/enterprise-java/jpa/persist-object-with-jpa/

https://www.baeldung.com/hibernate-entitymanager

https://www.tutorialspoint.com/de/jpa/jpa_entity_managers.htm

Ich habe Ihnen deshalb eine Anleitung verfasst, in der ich meine Erfahrungen und lauffähige Lösung zur Verfügung stelle.

Sie setzen dieses Beispiel um:

- Java Projekt mit Maven, Verwendung von JPA, Hibernate Session und Transaktionen.

⇨ Ablauf der Transaktion wird deutlich.
Unterstützung durch EntityManager fehlt noch.

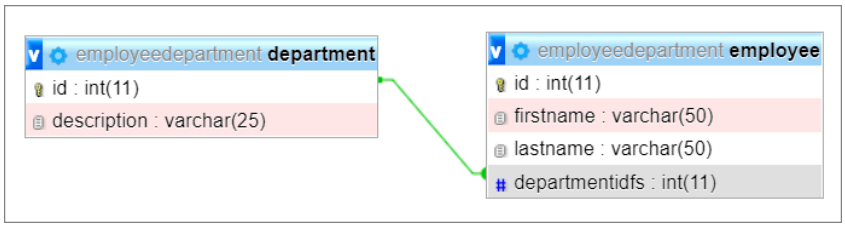
Ziele:

⇨ Sie können Entitäten aus einer Datenbank über das Persistenz-Framework und das Interface JPA mit Objekten der Applikation verwalten.

Inhaltsverzeichnis

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

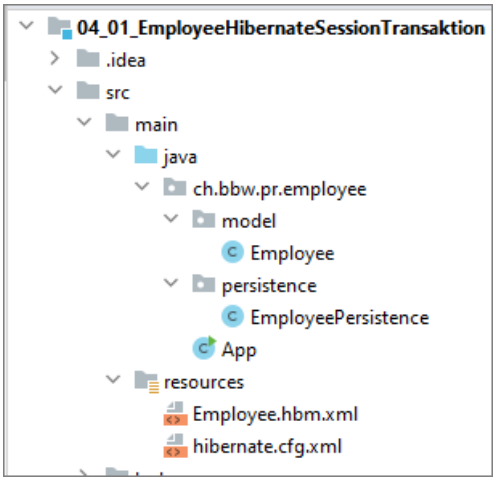
```
<!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.5.7.Final</version>
</dependency>
```

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

Employee.hbm.xml macht das Mapping zur Entität in der DB über eine Datei.

hibernate.cfg.xml ist die Konfigurationsdatei für den Datenbankzugriff

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

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

Übernehmen Sie die Klasse *EmployeePersistence.java*

Erklärungen:

```
20  public class EmployeePersistence {
21     private static SessionFactory factory;
22     private static Session sessionObj;
23
24     public EmployeePersistence(){
25        //Setup SessionFactory
26        Configuration config = new Configuration().configure();
27        config.addAnnotatedClass(ch.bbw.pr.employee.model.Employee.class);
28        //config.addResource("Employee.hbm.xml");
29        ServiceRegistry serviceRegistry = new StandardServiceRegistryBuilder()
30           .applySettings(config.getProperties()).build();
31        factory = config.buildSessionFactory(serviceRegistry);
32     }
33
34     public List<Employee> getAllEmployees(){
35        List<Employee> employees = null;
36        try{
37           //Transaction
38           sessionObj = factory.openSession();
39           Transaction tx = sessionObj.beginTransaction();
40           employees = (List<Employee>) sessionObj.createQuery("from Employee").getResultList();
41           tx.commit();
42
43        } catch(HibernateException e) {
44           if(null != sessionObj.getTransaction()) {
45              System.err.println("Transaction \"From Employee\" is Being Rolled Back.\n");
46              sessionObj.getTransaction().rollback();
47              e.printStackTrace();
48           }
49        } catch(Exception e) {
50           if(null != sessionObj.getTransaction()) {
51              System.err.println("Transaction \"From Employee\" is Being Rolled Back.\n");
52              sessionObj.getTransaction().rollback();
53              e.printStackTrace();
54           }
55        } finally {
56           if(sessionObj != null) {
57              sessionObj.close();
58           }
59        }
60        return employees;
61     }
62  }
```


Die Klasse behandelt den Zugriff auf die Datenbank mit Hilfe einer SessionFactory, Session, Transaction und einer Query.

Schlägt der Zugriff fehl, dann findet ein Rollback statt.

App.java

Implementieren Sie die App Klasse

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

App.java

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

import ch.bbw.pr.employee.persistence.EmployeePersistence;

/**
 * Applicatioin class
 * @author Peter Rutschmann
 * @version 17.09.2021
 */
public class App
{
    public static void main( String[] args )
    {
        System.out.println( "Employee with Hibernate Session and Transaction" );
        EmployeePersistence persistance = new EmployeePersistence();
        System.out.println("List all employees: " + persistance.getAllEmployees());
    }
}
```

Hibernate.cfg.xml

Achten Sie bei der Implementierung auf die Unterstützung Ihrer IDE!

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

hibernate.cfg.xml

```xml
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property name="hibernate.connection.driver_class">com.mysql.cj.jdbc.Driver</property>
        <property name="hibernate.connection.url">jdbc:mysql://127.0.0.1:3306/employeedepartment</property>
        <property name="hibernate.dialect">org.hibernate.dialect.MariaDB103Dialect</property>
        <property name="hibernate.connection.username">root</property>
        <property name="hibernate.connection.password">1234</property>
    </session-factory>
</hibernate-configuration>
```

Variante mit Mapping über Employee.hbm.xml für

Das Mapping zwischen Entität und Datenbank geschieht in der ersten Anwendung über die JPA Annotierungen in der Klasse Employee und durch die Zeile in der Klasse EmployeePersistence.

```
20  public class EmployeePersistence {
21      private static SessionFactory factory;
22      private static Session sessionObj;
23
24      public EmployeePersistence(){
25          //Setup SessionFactory
26          Configuration config = new Configuration().configure();
27          config.addAnnotatedClass(ch.bbw.pr.employee.model.Employee.class);
28          //config.addResource("Employee.hbm.xml");
```

Das Mapping kann auch über eine Konfigurations-Datei gemacht werden.

Dann muss allerdings *addAnnotatedClass* auskommentiert und *addResource* aktiviert werden.

Verstehen Sie die Datei?

Employee.hbm.xml

```
1   <?xml version='1.0' encoding='utf-8'?>
2   <!DOCTYPE hibernate-mapping PUBLIC
3           "-//Hibernate/Hibernate Configuration DTD//EN"
4           "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
5   <hibernate-mapping>
6       <class name="ch.bbw.pr.employee.model.Employee" table="employee">
7           <id name="id" type="int">
8               <column name="id" />
9               <generator class="identity" />
10          </id>
11          <property name="firstname" type="java.lang.String">
12              <column name="firstname" />
13          </property>
14          <property name="lastname" type="java.lang.String">
15              <column name="Lastname" />
16          </property>
17          <property name="departmentidfs" type="java.lang.Integer">
18              <column name="departmentidfs" />
19          </property>
20      </class>
21  </hibernate-mapping>
```

Für die Schnellen... probieren Sie es aus.