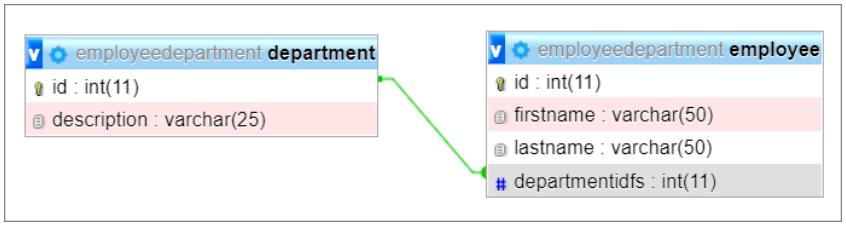
Umsetzung JPA, Hibernate mit Entity Manger, Beziehung

Einleitung:

Bisher haben Sie die Daten aus jeweils einer Tabelle der Datenbank gelesen.

Doch Employee steht mit Department in einer Beziehung.



Diese Beziehung c-mc kann mit Hilfe von JPA so aufgelöst werden, dass anstelle des Fremdschlüsselwertes eine Referenz auf das oder die Objekte der anderen Klasse verwaltet werden können.

Sie nutzen die Eigenschaften von JPA mit Beziehungen umzugehen.

- ⇨ 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.
- ⇨ Sie setzen **Beziehungen** für die Verwaltung der Daten ein.

Inhaltsverzeichnis

Input zur Beziehung mit JPA

Mit Hilfe von JPA können Fremdschlüsselwerte durch Referenzen auf Objekte aufgelöst werden.

Damit kann die Klasse *Employee* eine Referenz auf ein Department erhalten.
Anstelle des Fremdschlüsselwertes wird dann das Department sichtbar.

```
List all employees:
[Employee{id=3, firstname='hans', lastname='muster', department=Department{id=1, description='sales'}},
Employee{id=5, firstname='paula', lastname='kuster', department=Department{id=2, description='development'}},
 Employee{id=9, firstname='Lars', lastname='Lustig', department=null}]
```

Unterschiedliche Beziehungen können mit JPA und Hibernate elegant durch weitere ***Annotations*** konfiguriert werden.

Sie finden dazu auf dem Internet viele Anleitungen unter den Stichworten

Many-to-One mapping

One-to-Many mapping

Many-to-Many mapping

One-to-One Association

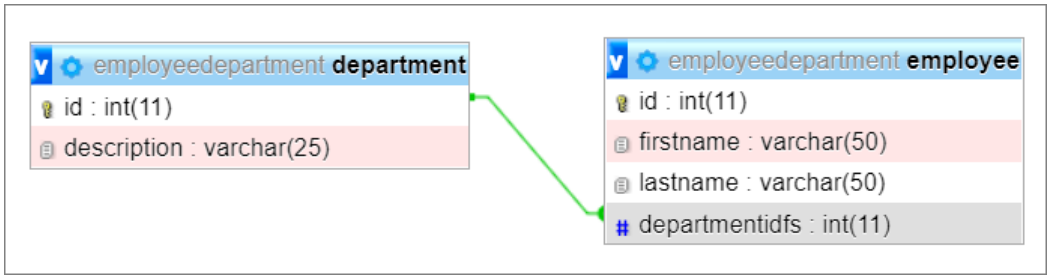
Hibernate Foreign Key

Zum Beispiel unter:

https://examples.javacodegeeks.com/enterprise-java/jpa/many-to-one-mapping-in-jpa/

Das Beispiel von javacodegeeks behandelt die Beziehung zwischen *Department* und *Employee*.

In meiner Implementierung habe ich nicht alle Attribute übernommen.



Beziehungen:

Aus Sicht der Entität *Employee* besteht eine **Many-To-One Beziehung** zur Entität *Department*.

Aus Sicht der Entität *Department* besteht eine **One-To-Many Beziehung** zur Entität *Employee*.

Oder man sieht das Ganze als eine ***Bidirektionale Beziehung*** **Many-To-One und One-To-Many.**

⇨ *Sie haben damit drei Möglichkeiten für die Umsetzung. Und dies sind unterschiedlich schwer.*

Nicht immer lassen sich die Beispiele auf Anhieb für das eigene Problem anwenden.
Es gibt unzählige Möglichkeiten von Fehlermeldungen 😊☹😐

Manchmal lohnt es sich, das Beispiel aus dem Tutorial zu implementieren und zum Laufen zu bringen. Und dann das eigene Beispiel davon abzuleiten. Schritt um Schritt…

Aufgabe Implementierung Many To One

Ergänzen Sie die Klasse Employee um die JPA Annotation für eine @ManyToOne Beziehung.

Damit ein Bezug zum Fremdschlüssel hergestellt wird, verwenden Sie zusätzlich die Annotierung @JoinCloumn

Das ursprüngliche Attribut departmentIDFS entfällt.

```
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      private int id;
18
19      @Column(name = "firstname")
20      private String firstname;
21
22      @Column(name = "lastname")
23      private String lastname;
24
25      //@Column(name = "departmentidfs")
26      //private Integer departmentidfs;
27
28      @ManyToOne
29      @JoinColumn(name = "departmentidfs")
30      private Department department;
31
32      public Employee() {
33      }
```

Sie müssen Getter und Setter bereinigen und die toString Methode überarbeiten.

Damit ein Employee ein Department hat, müssen Sie in der Datenbank bei den Employees Fremdschlüsselwerte für Departments ergänzen.

Als Ergebnis sehen sie dann die zu den Employees zugewiesenen Departments.

```
List all employees:
[Employee{id=3, firstname='hans', lastname='muster', department=Department{id=1, description='sales'}},
 Employee{id=5, firstname='paula', lastname='kuster', department=Department{id=2, description='development'}},
 Employee{id=9, firstname='Lars', lastname='Lustig', department=null}]
```

Aufgabe CRUD für angepasste Klasse Employee

Durch die Beziehung wird der Umgang mit CRUD für der Klasse Employee etwas komplexer.

1) Erzeugen Sie einen neuen Employee der einem existierenden Department zugeteilt wird.
   Sie müssen zuvor das Department aus der Datenbank abrufen und dann dieses Objekt dem neuen Employee zuweisen und dann den Employee in der Datenbank speichern.
   Geben Sie alle Employees zur Kontrolle aus.
   Kontrollieren Sie, ob der neue Employee erscheint.
2) Rufen Sie ein Department aus der Datenbank ab. Ändern Sie die *description* des Departments.
   Geben Sie alle Employees aus.
   Kontrollieren Sie, ob Sie die Änderung bei den Employees sehen.
3) Ändern Sie die *descriptions* wieder zurück.
   Geben Sie alle Employees aus.
   Kontrollieren Sie, ob Sie die Änderung bei den Employees sehen.
4) Löschen Sie den zuvor hinzugefügten Employee.
   Geben Sie alle Employees aus.
   Kontrollieren Sie, ob Sie die Änderung bei den Employees sehen.

Beispiel für Output:

```
Add new Employee with existing Department
Department{id=2, description='Development'}
List all employees:
[Employee{id=1, firstname='Lars', lastname='Lustig', department=null}, Employee{id=3,
 firstname='hans', lastname='muster', department=Department{id=1, description='sales'}},
 Employee{id=18, firstname='Niklaus', lastname='Waldmann', department=Department{id=2,
 description='Development'}}]

Update department Development to Entwicklung
List all employees:
[Employee{id=1, firstname='Lars', lastname='Lustig', department=null}, Employee{id=3,
 firstname='hans', lastname='muster', department=Department{id=1, description='sales'}},
 Employee{id=18, firstname='Niklaus', lastname='Waldmann', department=Department{id=2,
 description='Entwicklung'}}]

Update department back to Development:
List all employees:
[Employee{id=1, firstname='Lars', lastname='Lustig', department=null}, Employee{id=3,
 firstname='hans', lastname='muster', department=Department{id=1, description='sales'}},
 Employee{id=18, firstname='Niklaus', lastname='Waldmann', department=Department{id=2,
 description='Development'}}]

Delete Employee:
List all employees:
[Employee{id=1, firstname='Lars', lastname='Lustig', department=null}, Employee{id=3,
 firstname='hans', lastname='muster', department=Department{id=1, description='sales'}}]
```

Problem?

Ich bekam diese Fehlermeldung:

A different object with the same identifier value was already associated with the session

Geholfen hat mir dieser Hinweis, den ich mit Hilfe von Google gefunden habe:

My solution is to add this code on your primary key:

```
@GeneratedValue(strategy = GenerationType.AUTO)
```