bbw-it wiki

Hibernate HQL Abfragen

Für Zugriffe auf die Datenbank wird anstatt SQL HQL verwendet (Hibernate Query Language). Der Hauptunterschied ist der, dass bei HQL mit Klassennamen gearbeitet wird statt mit Tabellennamen.

Beispiele

Alle Beispiele beziehen sich auf die Filmclub-Datenbank, die Sie aus einem Projekt im Modul 151 kennen.

Alle Beispiele gehen davon aus, dass in der Variable `session` ein gültiges Session-Objekt referenziert wird.

Laden eines Objekts mit bekannter ID

Variante 1:

```
Presentation p = (Presentation)session.get(Presentation.class, 1);
```

Variante 2:

```
Presentation p = new Presentation();
session.load(p, 1); // Load the presentation object with ID=1 into p
```

Laden einer ungefilterten Liste

```
List<Member> members = session.createQuery("FROM Member").list();
for (Member member : members) {
    System.out.println(member.getFirstName() + " " + member.getLastName() + ": " +
                member.getPresentations().size() + " Besuche");
    for (Presentation p : member.getPresentations()) {
        System.out.println("    " + p.getFilm().getTitle());
    }
}
```

Wie man an diesem Beispiel sieht, hat man, nachdem ein Objekt geladen wurde, auch Zugriff auf die verknüpften Objekte (in diesem Fall von Member auf Presentation), ohne dass dies extra programmiert werden müsste. Dies kann man so einstellen, dass die verknüpften Objekte von Anfang an mitgeladen werden (eager mode) oder erst, wenn auf sie zugegriffen wird (lazy mode).

Laden einer Liste mit Filterkriterien

Es werden alle Member-Objekte geladen, deren Name mit 'M' beginnt und die an einer Vorführung eines Films waren, dessen Titel den String 'ERIN' enthält.

```
List<Member> filteredMembers = session.createQuery(
        "FROM Member m inner join fetch m.presentations  as p " +
        " WHERE m.lastName LIKE 'M%' AND p.film.title LIKE '%ERIN%'").list();
for (Member member : filteredMembers) {
    System.out.println(member.getFirstName() + " " + member.getLastName() + ": " +
            member.getPresentations().size() + " Besuche");
    for (Presentation p : member.getPresentations()) {
        System.out.println("    " + p.getFilm().getTitle());
    }
}
```

Beachten Sie hier, wie auch der JOIN sich auf Klassen und Objekte bezieht, nicht auf Tabellen.

Update eines bestehenden Objekts

```
Transaction t = session.beginTransaction();
Member m1 = (Member)session.get(Member.class, 5); (( get Member object with ID 5
m1.setLastName("00UpdatedName"); // Change its name
session.update(m1); // and store it
t.commit();
```

Beachten Sie hier auch, dass Updates innerhalb einer Transaktion erfolgen müssen. Bis zum Aufruf vom `commit` auf der Transaktion wird an der Datenbank nichts definitiv verändert.

Erstellen und speichern eines neuen Objekts

```
Member m2 = new Member();
m2.setFirstName("0Demo4");
m2.setLastName("0Demo4");
Transaction t = session.beginTransaction();
System.out.println("ID=" + m2.getMemberId());
session.save(m2);
System.out.println("ID=" + m2.getMemberId());
t.commit();
```

</code>

Beachten Sie hier, dass bei ersten `println` die ID noch `null` ist, während sie beim zweiten `println` einen gültigen Wert bekommen hat.

Einen Listeneintrag zu einem Objekt hinzufügen

Der folgende Code fügt einem Member einen Vorstellungsbesuch hinzu und speichert dies so auf der Datenbank.

```
Member m = new Member();
session.load(m, 2);

Transaction t2 = session.beginTransaction();
m.getPresentations().add(p);
session.update(m);
t2.commit();
session.close();
```
