链滴

Hibernate 学习笔记（三）

作者：chillax
原文链接：https://ld246.com/article/1484997745252
来源网站：链滴

Hibernate学习笔记（一对多、多对多操作）

ibernate一对多映射配置

    1.
先创建两个实体类，我便以客户与联系人为例子说明

    2.
个实体类之间要相互的表示

nbsp;(1)　　**在客户实体类里面有多个联系人**

```
//hibernate 中要求使用set集合来存储类似这种情况
private Set<LinkMan> setLinkMan = new HashSet<LinkMan>();
```


nbsp;(2)　　**一个联系人属于一个客户**

```
//表示联系人所属于的客户
private Customer customer;
```

    3.
射文件里面配置一对多的关系

nbsp;(1)　　**客户映射文件中表示所有联系人,使用set标签表示多个联系人，其中name属性填写的一对多关系中多的一方集合的名字，即填写客户实体类里面客户set集合的名称,而里面的key标签是名数据库表的外键名称，<one - to-many>标签是指明两个实体类的关系**
一对多、class属性填写的是多的一方的实体类路径

```
<!-- 在客户的映射文件中表示所有联系人 -->
<set name="setLinkMan">
 <!-- 一对多建表，有外键
    Hibernate双向维护外键，两方都要配置
    column属性值是外键名称
 -->
   <key column="clid"></key>
   <!-- 客户所有的联系人，class里面写联系人实体类全路径 -->
   <one-to-many class="entity.LinkMan"/>
</set>
```


nbsp;(2)　　**在联系人实体类里面表示所属的客户,同样，<any-to-one>是指明两个实体类的关系，其中class指明一的那一方实体类的全路径，而column是要填写外键的名称，也就是之前在客户实体类配置文件里填写的外键**

```
<!-- 联系人所属的客户 -->
<many-to-one name="customer" class="entity.Customer" column="clid">
```

    4.
核心配置文件中将两个映射文件引入

ibernate一对多级联操作

    1.
联保存（例如：添加一个客户，为这个客户添加联系人）

nbsp;(1)　　**复杂做法，需要在两个实体类之间相互表示关系，对两张表同时更新**

```
//创建客户的对象
Customer customer = new Customer();
customer.setCustName("it");
customer.setCustLevel("vip");
customer.setCustSource("net");
customer.setCustPhone("110");
customer.setCustMobile("999");
//创建联系人对象
LinkMan linkMan = new LinkMan();
linkMan.setLkm_name("luck");
linkMan.setLkm_gender("男");
linkMan.setLkm_phone("121");
//建立两个实体类之间的关系
customer.getSetLinkMan().add(linkMan);
linkMan.setCustomer(customer);
//向数据库中存储
session.save(linkMan);
session.save(customer);
```


nbsp;(2)　　**简化做法，首先需要在多的一方的映射文件中配置，在其set标签里加一个属性，cascade属性，它的值有save-update、delete等，分别表示级联更新、级联删除，多个属性之间用逗号开如：“save-update，delete”。这时候只需要将联系人加入到客户里面、更新客户即可**

```
<set name="setLinkMan" cascade="save-update">

customer.getSetLinkMan().add(linkMan);
session.save(customer);
```

    2.
联删除（例如：删除一个客户，这个客户里面的所有联系人也要删除）

nbsp;(1)　　**需要在配置文件里面配置，参照上文**

```
<set name="setLinkMan" cascade="save-update,delete">
```


nbsp;(2)　　**在代码中就可以直接删除客户了**

```
Customer customer = session.get(Customer.class, 1);
session.delete(customer);`
```


ibernate中一对多的修改操作

    1.
接修改会产生一个问题，需要配置一个属性inverse="true"，因为Hibernate对外键是双向维护，对外键进行两次更新，降低了效率，因此将inverter属性设置为true可以让一的一方放弃维护。

```
Customer baidu = session.get(Customer.class, 4);
LinkMan linkMan = session.get(LinkMan.class, 3);
baidu.getSetLinkMan().add(linkMan);
linkMan.setCustomer(baidu);
```


ibernate的多对多操作（例:用户和角色）

    1.
对多的映射配置

nbsp;(1) **创建实体类，用户和角色**

```
User
private Integer user_id;
private String user_Name;
private String user_password;
Role
private Integer role_id;
private String role_name;
private String role_memo;
```


nbsp;(2) **一个用户表示多个角色**

```
private Set<Role> setRole = new HashSet<Role>();
```


nbsp;(3) **一个角色有多个用户**

```
private Set<User> setUser = new HashSet<User>();
```


nbsp;(4) **配置多对多关系，与一对多关系中多的一方配置相似**

```
角色的配置文件
<set name="setUser" table="user_role" cascade="save-update">
    <!-- 当前映射文件，在第三张表中的外键名称 -->
    <key column="roleid"></key>
    <!--
        class属性：角色实体类全路径
        column属性：角色在第三张表的外键名称
    -->
    <many-to-many class="entity.User" column="userid"></many-to-many>
</set>
```

```
用户的配置文件
<!-- 在用户里面表示所有的角色
      name属性:角色set集合的名称
      table属性:第三张表的名称
-->
<set name="setRole" table="user_role" cascade="save-update">
   <!-- 当前映射文件，在第三张表中的外键名称 -->
   <key column="userid"></key>
   <!--
      class属性：角色实体类全路径
      column属性：角色在第三张表的外键名称
    -->
   <many-to-many class="entity.Role" column="roleid"></many-to-many>
</set>
```


nbsp;(5) **在核心配置文件导入映射文件**

    2.
对多的级联保存

nbsp;(1) **根据用户保存角色，在用户映射文件里面讲set标签里的cascade属性设置为save-update**

```
//建立用户
User user1 = new User();
User user2 = new User();
//建立角色
Role r1 = new Role();
Role r2 = new Role();
Role r3 = new Role();
//user1有r1\r2的角色
user1.getSetRole().add(r1);
user1.getSetRole().add(r2);
//user2有r2\r3的角色
user2.getSetRole().add(r2);
user2.getSetRole().add(r3);
//保存用户
session.save(user1);
session.save(user2);
```

    3.
对多的级联删除一般不会去用，因为会导致用户与角色表的数据删除，因此不需要级联删除，只需要维护第三张关系表就可以了

    4.
护第三张表

nbsp;(1) **让用户有某些角色，直接将用户和角色查出来，在用户的角色集合中添加即可**

```
User user = session.get(User.class, 3);
Role role = session.get(Role.class,1);
user.getSetRole().add(role);
```


nbsp;(2) **让用户没有某些角色，也是直接将用户和角色查出来，在用户的角色集合中删除即可**

```
User user = session.get(User.class, 3);
Role role = session.get(Role.class,1);
user.getSetRole().remove(role);
```