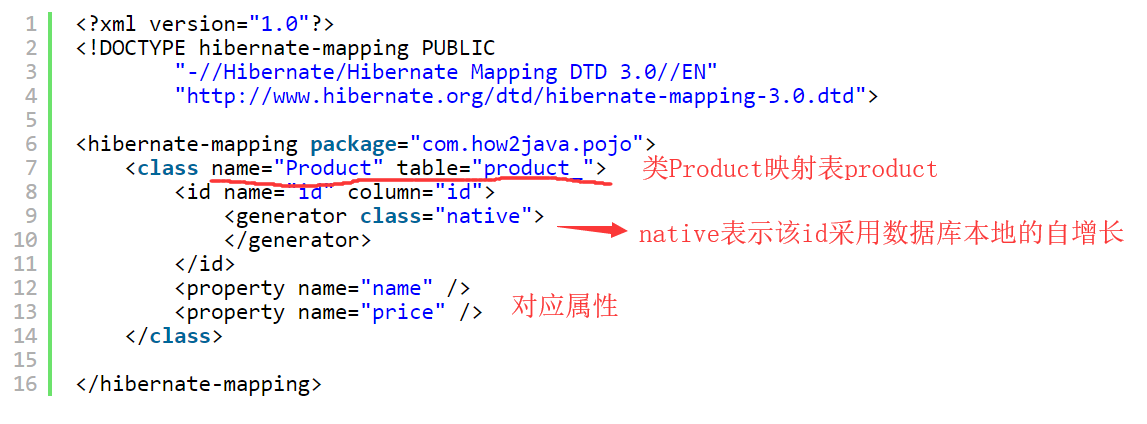
## 配置数据库的映射：

在包com.how2java.pojo下 新建一个配置文件Product.hbm.xml， 用于映射Product类对应数据库中的product\_表

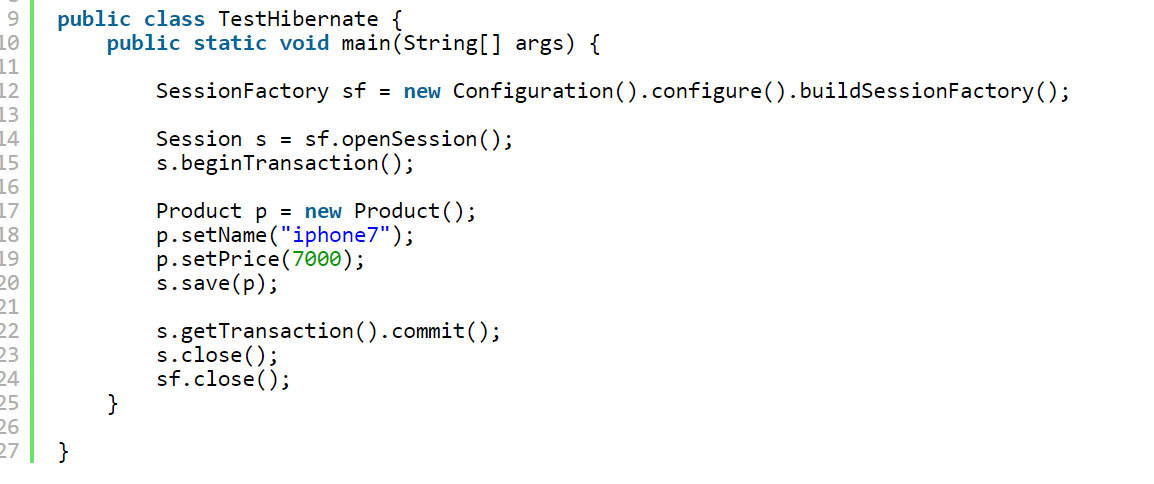


## 配置数据库和hibernate的链接config：

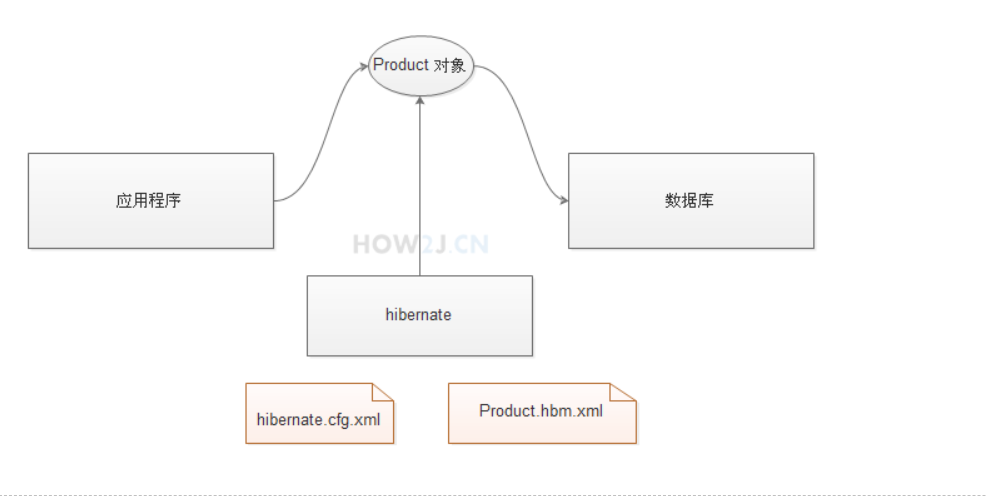
在**src目录**下创建 hibernate.cfg.xml  
配置访问数据库要用到的驱动，url，账号密码等等

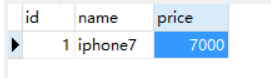


然后就可以写入数据库了：



这里的Session相当于连接数据库的一个线程。“数据库线程”



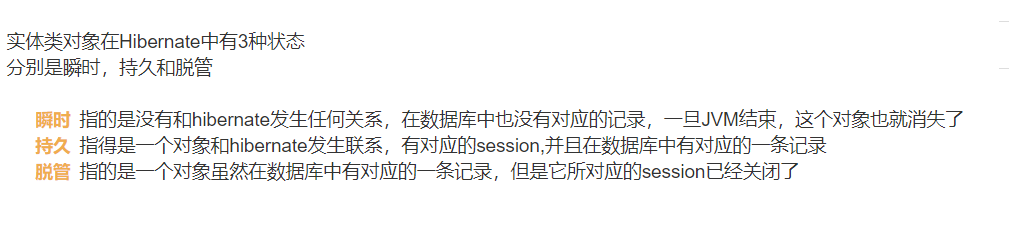


总的来说一共需要三个配置文件

1. 实体类
2. Xml映射
3. Hibernate-config文件

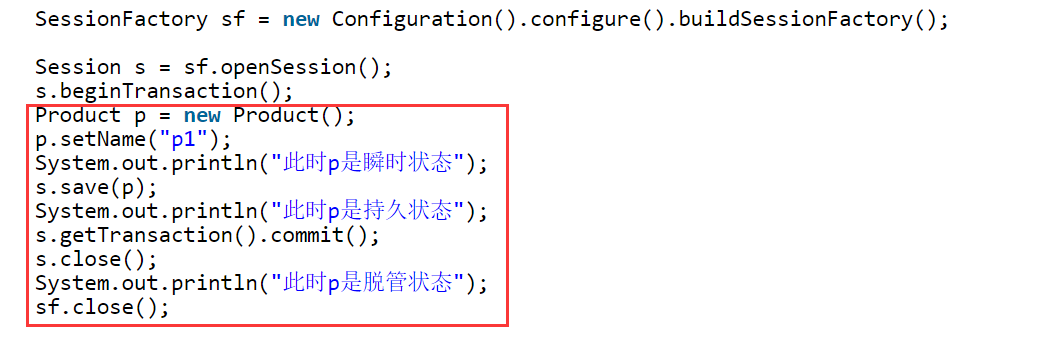
就可以完成整个的hibernate流程了。

## 3.对象状态：



new 了一个Product();，在数据库中还没有对应的记录，这个时候Product对象的状态是**瞬时的**。  
  
通过Session的save把该对象保存在了数据库中，该对象也和Session之间产生了联系，此时状态是**持久的**。  
  
最后把Session关闭了，这个对象在数据库中虽然有对应的数据，但是已经和Session失去了联系，相当于脱离了管理，状态就是**脱管的**

就是使用hibernate之前，之中，之后的状态。



## 4.增删改查

很简单的操作，以对象为例即可进行操作即可。

## 5.外联表

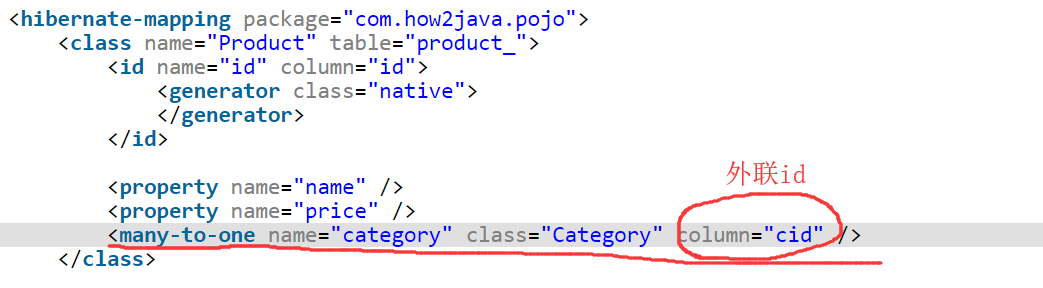
### 1.多对一的关系：

一个product对应一个category

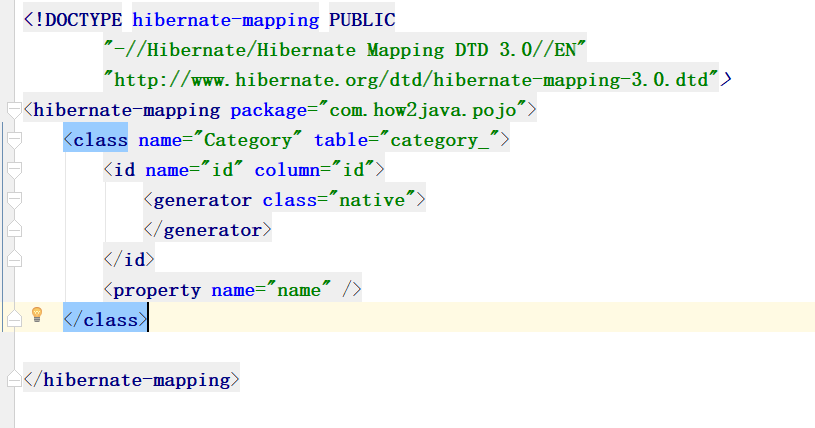
一个category对应对个product

Product表要外联category的id

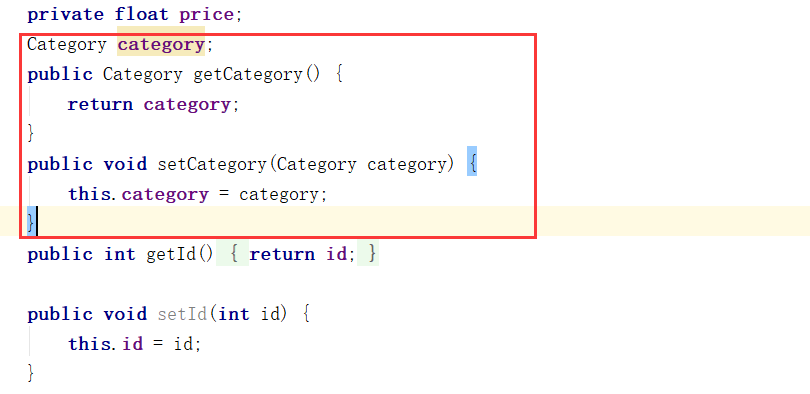
可以在product中设置：

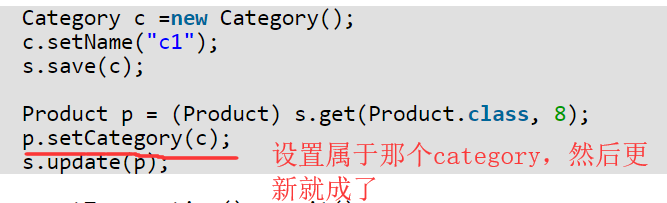


这是category表的样子：



还有要在product类中添加category的类：

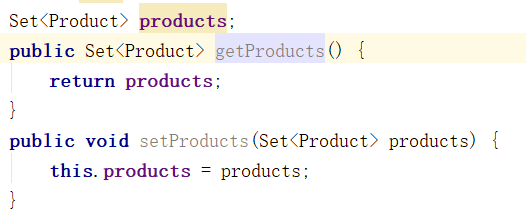




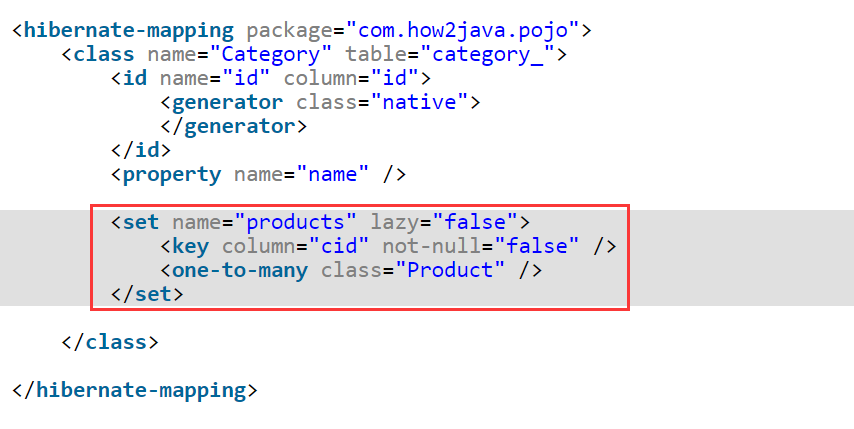
### 2.一对多的关系

Category对应product是一对多的关系。

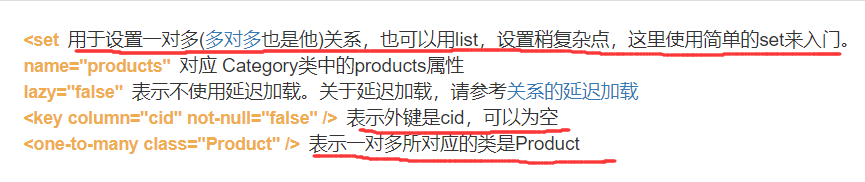
1. 先为category类添加一个set，表示内置多个不同的product



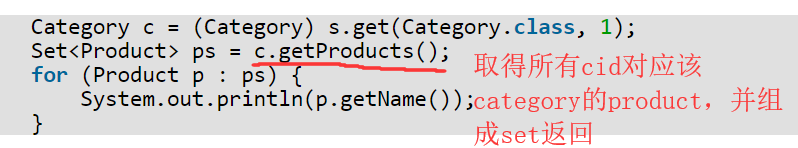
1. 在xml配置中设置一对多关系：



解释：



使用：



### 多对多的关系：

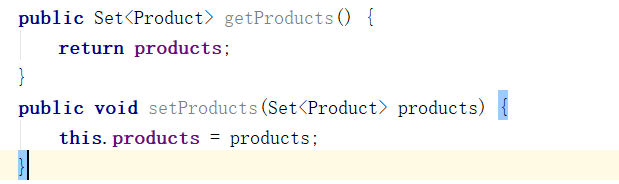
一个user可以购买多个product

一个product可以被多个user购买

所以product和user是多对多的关系。

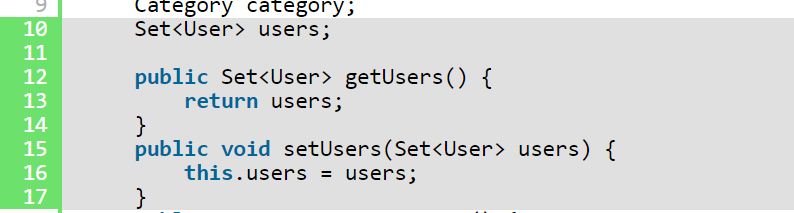
实现多对多必须有一个中间表去记录（和原来一样）

1. 定义user类，在其中设置一个user对应多个product：



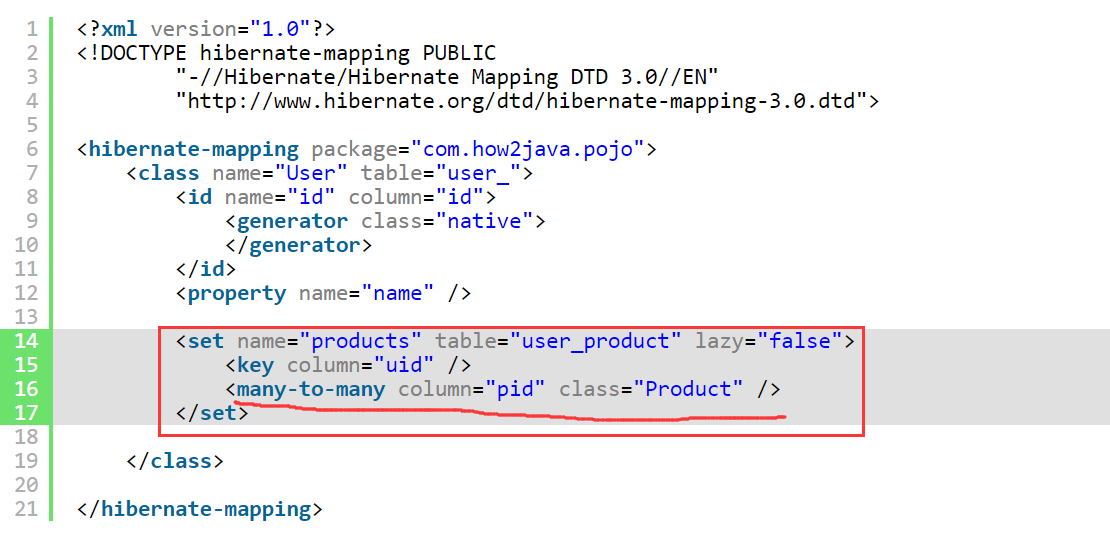
Product中同理设置：

多对多关系：

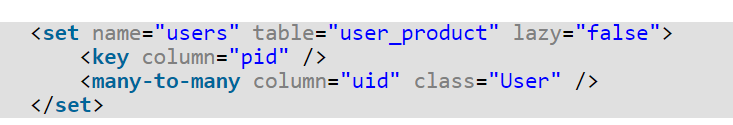


1. Xml中设置多对多：

User表中：



Product表中：

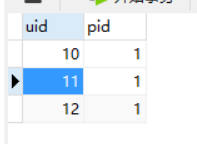


注意这里面的set 中name 对应的是类中的属性，比如User类中的Product属性，Product类中对应的user属性，因为是多对多，他们都会集中在user\_product这个table当中（table设置）。而且他们中对应的key值，many-to-many中的column都是互相相对的

3.多对多购买：



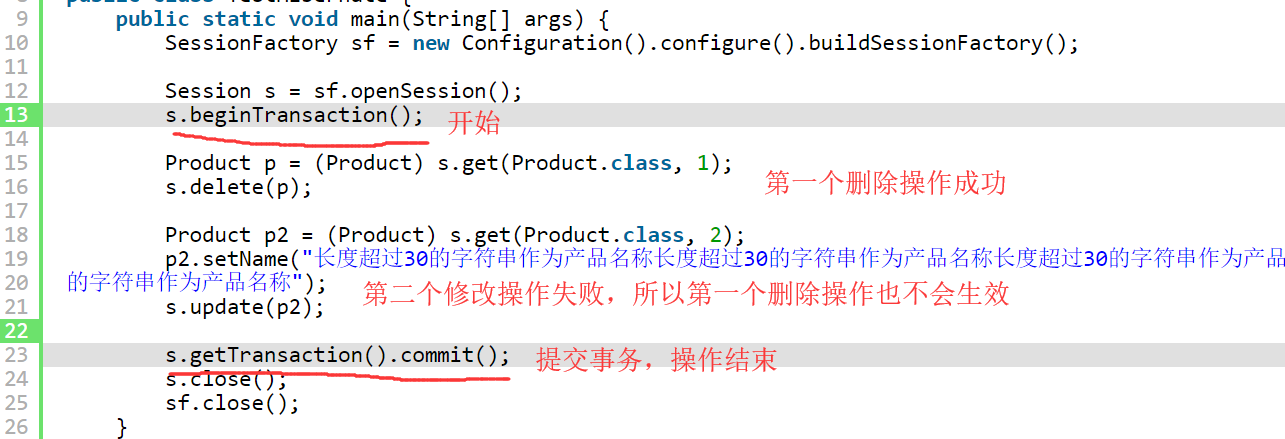
结果：



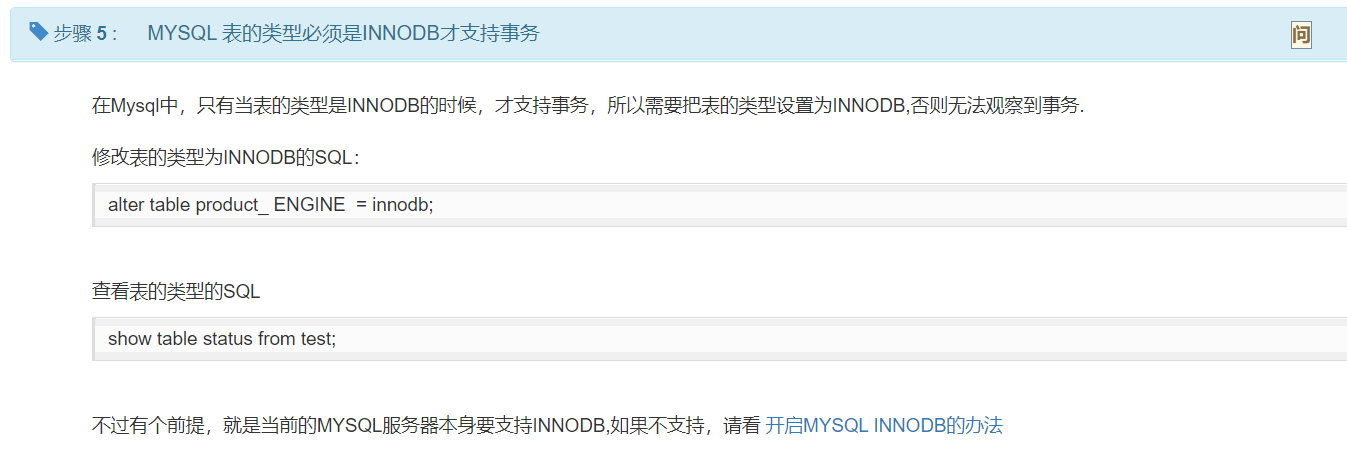
## 6.事务：

Hibernate中的任何操作都应该放在事务当中，要么都成功，要么都失败。

hibernate中的事务由s.beginTransaction();开始  
由s.getTransaction().commit();结束



数据库要支持innodb才可以进行事务处理。



## 7.load延迟访问方法（性能改善）：

这种延迟查询的方法都是性能改善的方法，

1.Load方法是先不查，只是打开通路，想要数据的时候再查的思想。

2.关系延迟加载是先查到必须的，至于关联的外部类的信息，到有想要的东西时，再进行查询的思想。

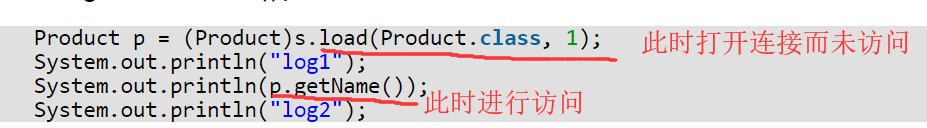
不同于get方法，get方法在使用的同时即执行查询语句。

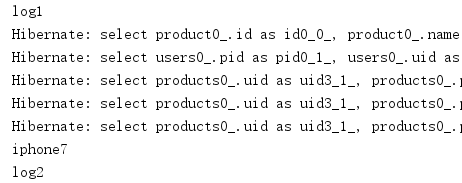
而load方法是延时进行的，就只是表示开启了对数据库的链接，而并没有执行数据访问，再需要使用数据的时候，才进行数据库的访问，达到了延时的效果。

原文:

load方式是延迟加载，只有属性被访问的时候才会调用sql语句  
get方式是非延迟加载，无论后面的代码是否会访问到属性，马上执行sql语句

都通过id=500去获取对象  
1. get方式会返回null  
2. load方式会抛出异常

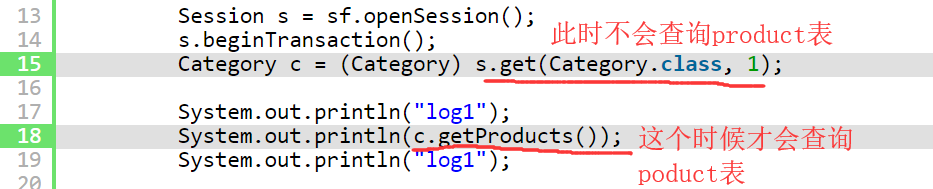


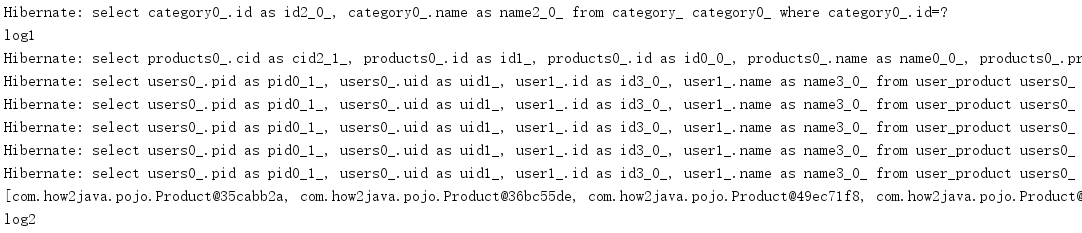


还有一种叫做关系延迟加载：

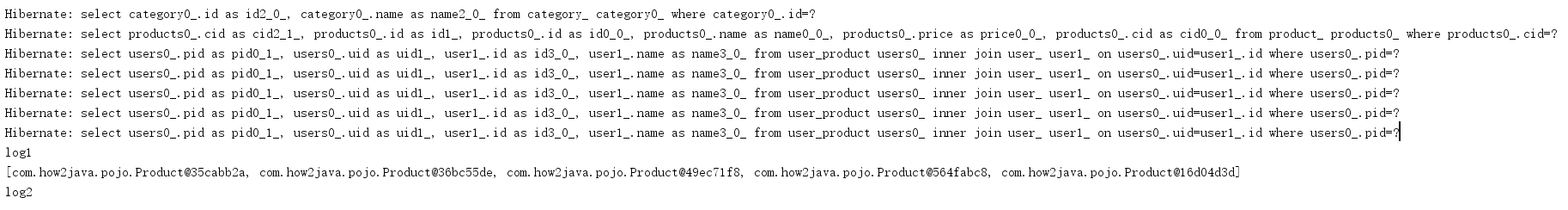
延迟加载又叫lazyload，在one-many many-many的时候都可以使用关系的延迟加载

即有表外信息的时候先不查询表外类的外联表，比如category有product的信息，那么在查询一个category时就不会先查询product，等需要使用category的关于product的内容时才会查询product表。





如果lazy=false就会这样： 先就会全部查询好。



## 8.级联

等级之间的连接，比如一个category被删了，如果设置了级联，那么这个category下面的所有product都会被删掉。

4种级联的设置：

包括上一步说的删除用得级联，级联有4种类型：  
all：所有操作都执行级联操作；  
none：所有操作都不执行级联操作；  
delete：删除时执行级联操作；  
save-update：保存和更新时执行级联操作；  
级联通常用在one-many和many-to-many上，几乎不用在many-one上

（不可能删除一个product而导致整个category被删掉。）。

设置：

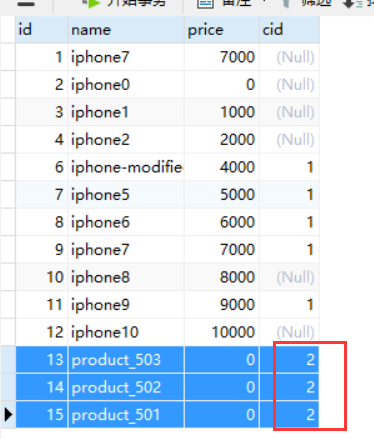


表示在一个category删除之后，其属于这个category的所有的product都将被删除掉。

Save-update的级联：

在保存或者更新的时候能够自动关联到对应表，比如新的product放到了某个category，提交之后这个product的cid会自动变成那个category的cid，不用手动设置。



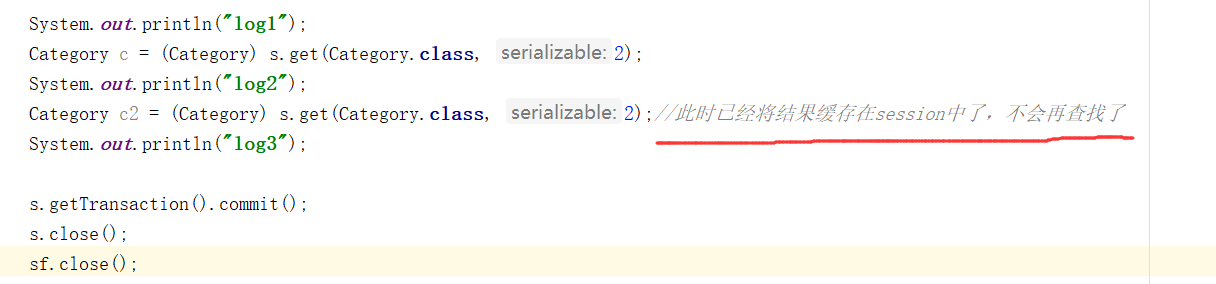


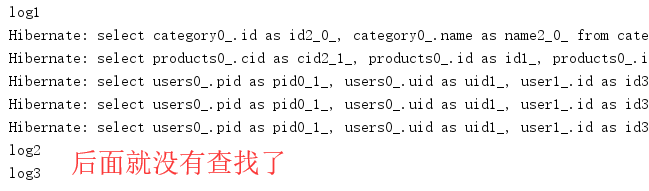
all 就是 delete+save-update  
none 就是没有，默认就是none

## 9.一级缓存（session中）

hibernate默认是开启一级缓存的，一级缓存存放在session上

在查找同一个对象时：





## 10.二级缓存（sessionFactory中）

Hibernate的一级缓存是在Session上，二级缓存是在SessionFactory上

也就是放在更上面一层了，有时候sessionA保存了c，但是另一个sessionB没有保存c，那么SessionB就又会查找一遍c

这时候就可以开启二级缓存，将c放在更上面一层的SessionFactory上面。

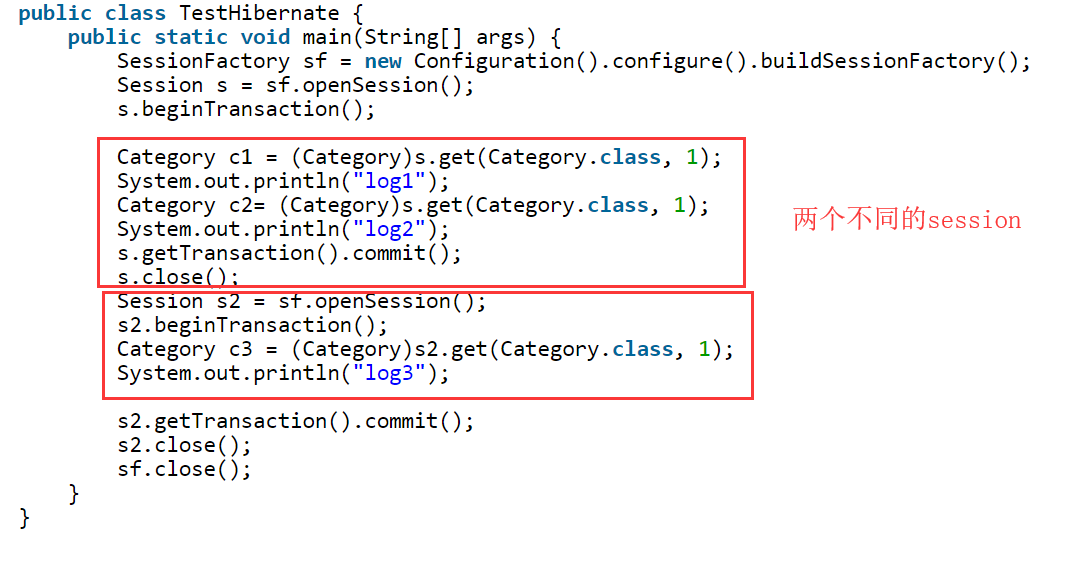
这样即使sessionB没有查找过c，也可以通过sessionFactory找到c了

原文：

使用不同的session,都去获取id=1的category,只会访问一次数据库。因为第二次获取虽然没有从第二个session中拿到缓存，但是从sessionfactory中拿到了Category缓存对象

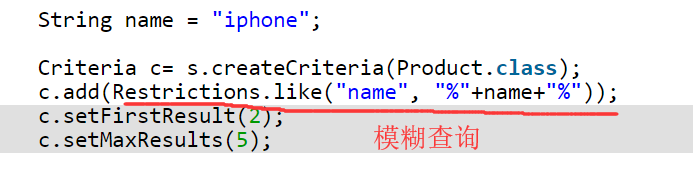
开启二级缓存的配置有点麻烦，见网页：

<https://how2j.cn/k/hibernate/hibernate-level-2-cache/47.html#nowhere>



## 11.分页：

使用Criteria进行分页，找出第一个和查询个数。（也就是limited）



**c.setFirstResult(2);** 表示从第3条数据开始  
**c.setMaxResults(5);** 表示一共查询5条数据

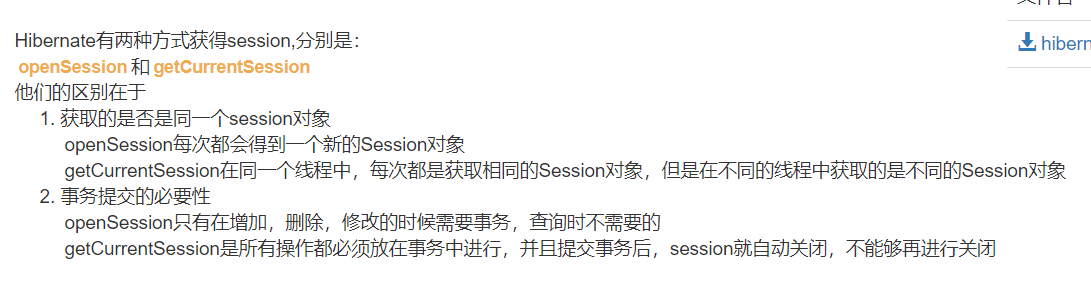
## 12.getCurrentSession vs openSession

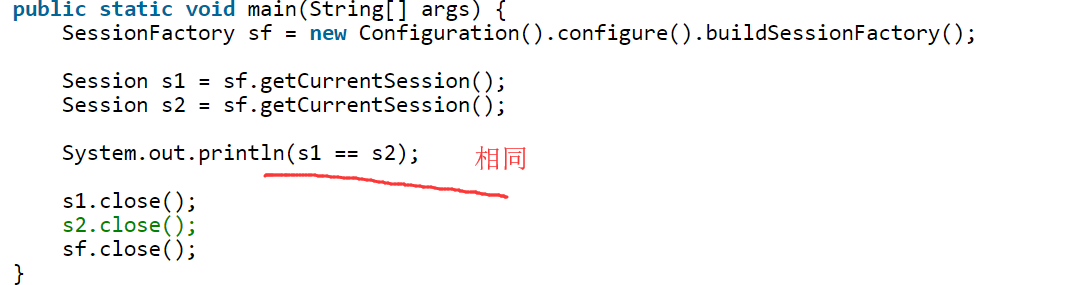
openSession就是平常玩的那种，他的增删改都必须放在事务当中，而查找不需要。

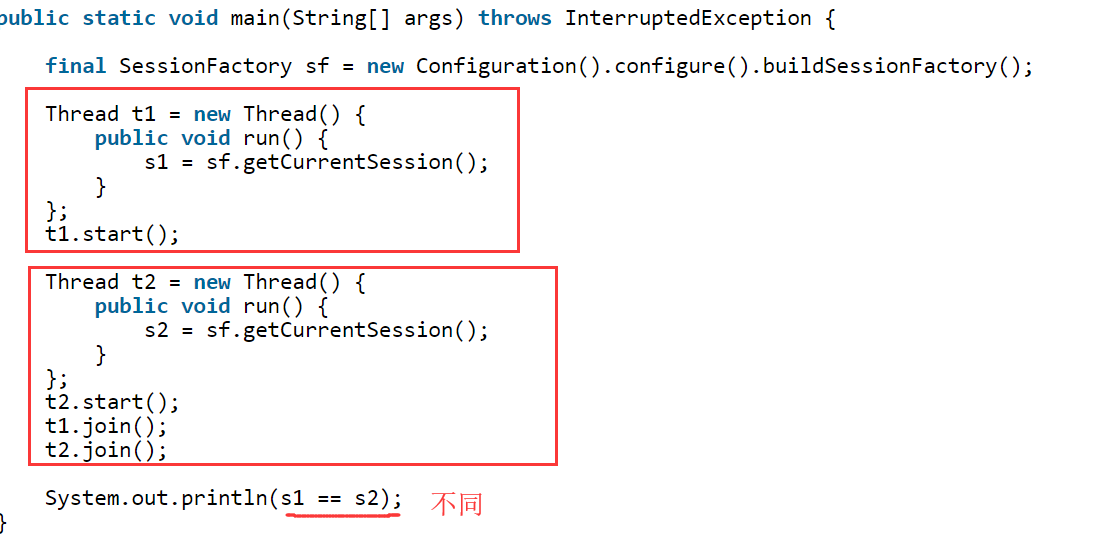
但是getCS就必须所有的增删改查都在事务当中。

在一个线程当中，不同的openSession是不同的内存地址，而getCS是同一个Session，如果在不同的线程当中的话，getCS就是不同的Session。

而且getCS是自动关闭的，不用手动close







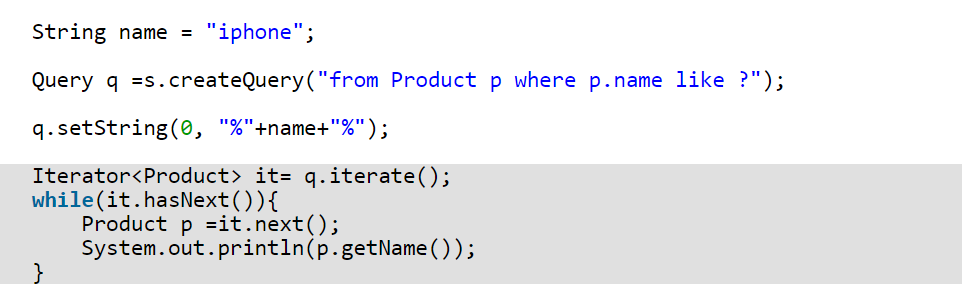
## 13.N+1查询（性能提升）

因为缓存机制的存在，在查找已经查过的数据（放在了缓存中的）会很快，但是如果查找100个实例，30个是已经查过了的，去查找的话还是会把100个重新查一遍，无视已经查过的30个，但是如果通过设置id，可以避免30个重复查询。

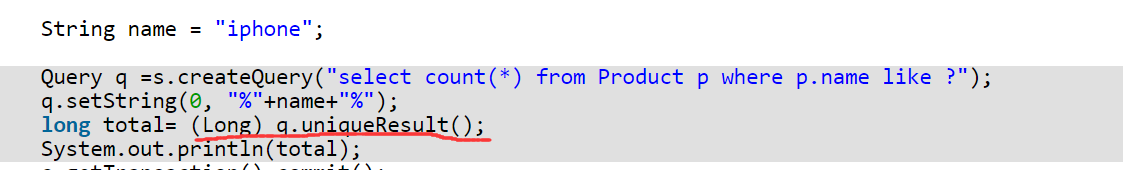
原文：

Hibernate有缓存机制，可以通过用id作为key把product对象保存在缓存中  
  
同时hibernate也提供Query的查询方式。假设数据库中有100条记录，其中有30条记录在缓存中，但是使用Query的list方法，就会所有的100条数据都从数据库中查询，而**无视这30条缓存中**的记录  
  
N+1是什么意思呢，首先执行一条sql语句，去查询这100条记录，但是，

只返回这100条记录的ID  
然后再根据id,进行进一步查询。  
  
如果id在缓存中，就从缓存中获取product对象了，否则再从数据库中获取



## 14.查询总数



## 15.乐观锁避免“插队”（version判断）

现在来看一个session相当于一个连接“数据库线程”

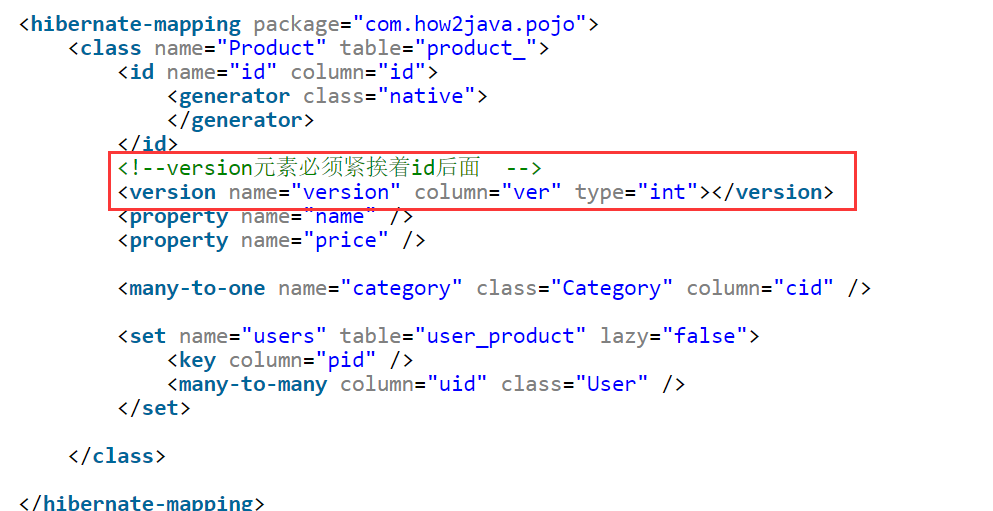
线程访问数据，多线程多session的情况下肯定会出现经典的“插队问题”，导致脏数据。



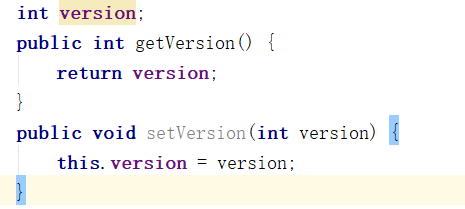
添加<version>到product的配置中，verson字段用来控制数据版本，以防插队情况。

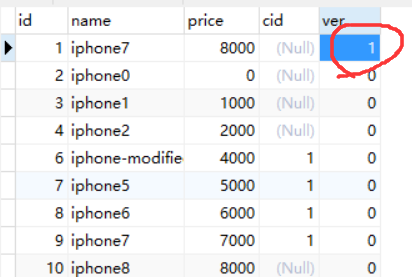
原文：

增加一个version字段，用于版本信息控制。这就是乐观锁的核心机制。  
  
比如session1获取product1的时候，version=1。 那么session1更新product1的时候，就需要确保version还是1才可以进行更新，并且更新结束后，把version改为2。



并在product类中添加version字段：

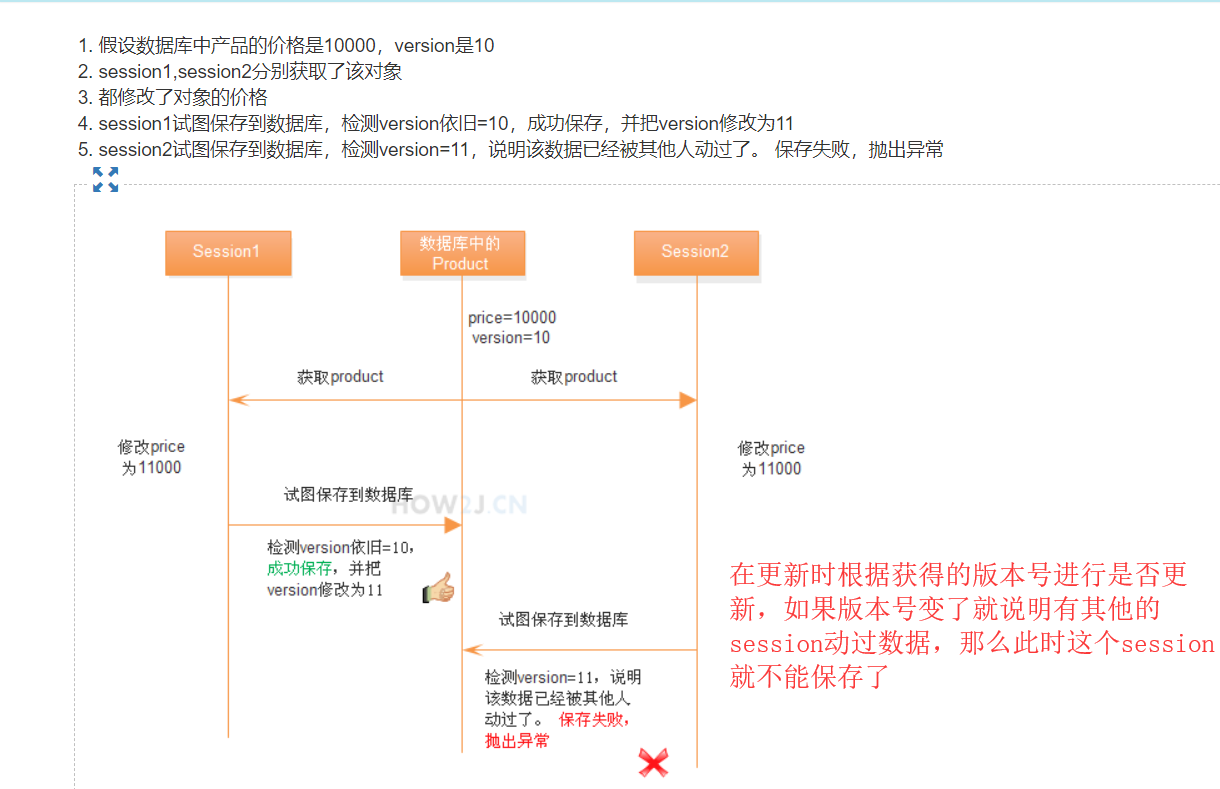




如果不同的session（数据库线程）做同样的操作导致version不一致就会报错，避免插队情况。



原理:



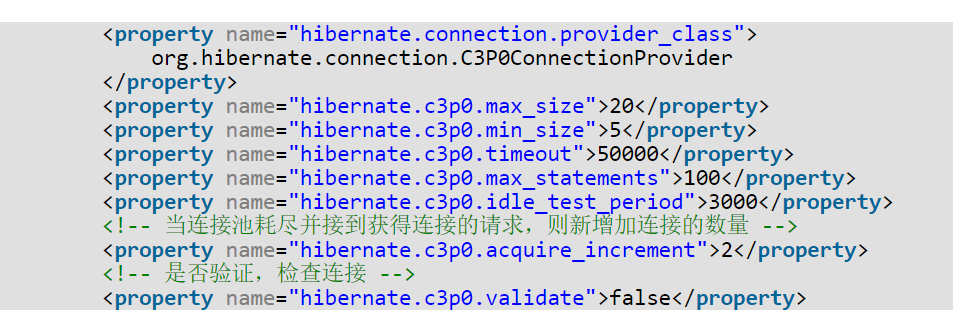
在更新时根据获得的版本号判断是否更新，如果版本号变了就说明有其他的session早先就动过数据了，那么此时这个session就不能保存更新的数据了。（在获得数据的时候得到一个version，在更新数据时用这个version和数据现在的version对比，看是否能更新）

## 16.hibernate连接池

连接池的好处就是减少多次数据库连接，一步到位的作用来提高性能。

Hibernate的连接池是处理高并发用的，这里只是教怎么配置，实际效果看不怎么出来。

建立数据库连接时比较消耗时间的，所以通常都会采用数据库连接池的技术来建立多条数据库连接，并且在将来持续使用，从而节约掉建立数据库连接的时间  
hibernate本身是提供了数据库连接池的，但是hibernate官网也不推荐使用他自带的数据库连接池。  
一般都会使用第三方的数据库连接池  
C3P0是免费的第三方的数据库连接池，并且有不错的表现



完整教程在：

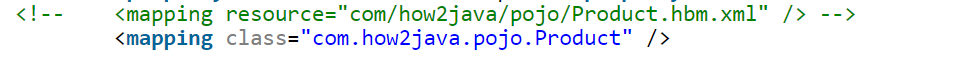
<https://how2j.cn/k/hibernate/hibernate-threadpool/48.html#nowhere>

## 17.使用注解的方式生成module

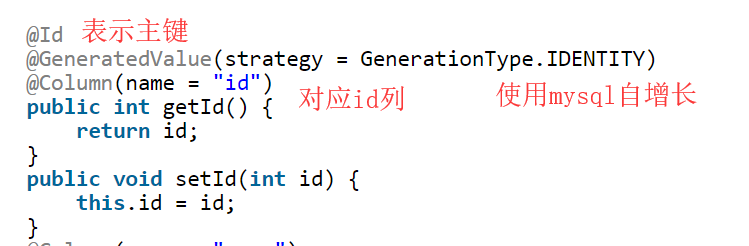
### 1.使用注解

终于有注解的方式可以玩玩了：

配置：







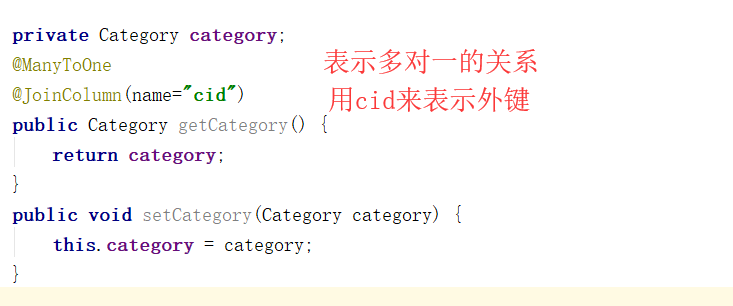


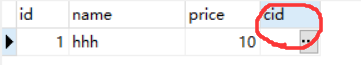
注意是放在get上面，不是set上面（上面红字有误）

### 2.多对一的设置：

Product和category是多对一：

在product类中表示：

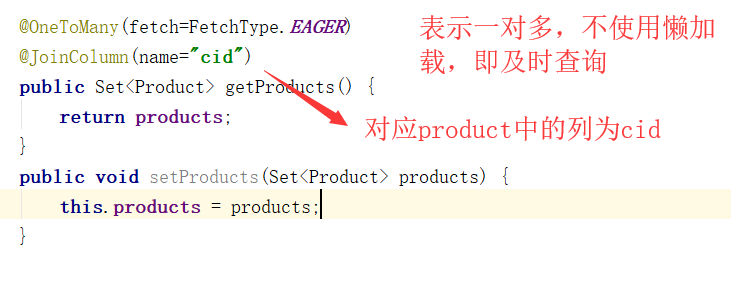




### 3.一对多的设置：

Category对于product是一对多的关系，要在category中添加set设置。

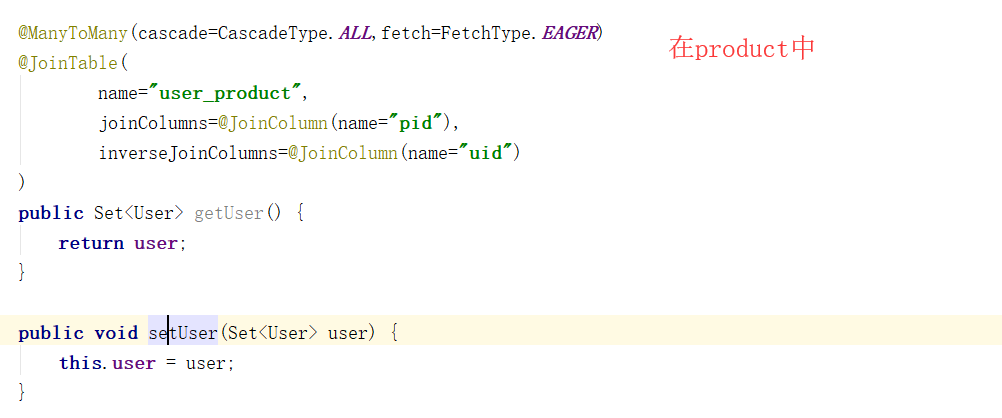
Category设置如下：



### 4.多对多的设置：

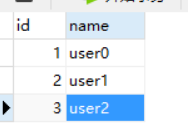
Product和user是多对多的设置：

注解的方式如下：





结果：



### 5.注解使用大全：

基本的使用方法就是上面的内容了，更多的内容在

<https://how2j.cn/k/hibernate/hibernate-annotation-manual/1051.html>



注解使用的规范？：

小项目，参与人数不多，不复杂的用注解，开发快速。 复杂项目，多人交互，配置量大，维护复杂度高的，用配置文件。

同意。