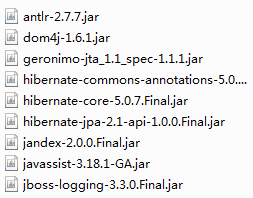
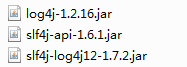
## Hibernate基础（HIbernate操作的都是类对象）

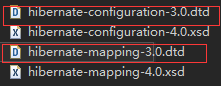
**一．核心配置文件**

1.导包

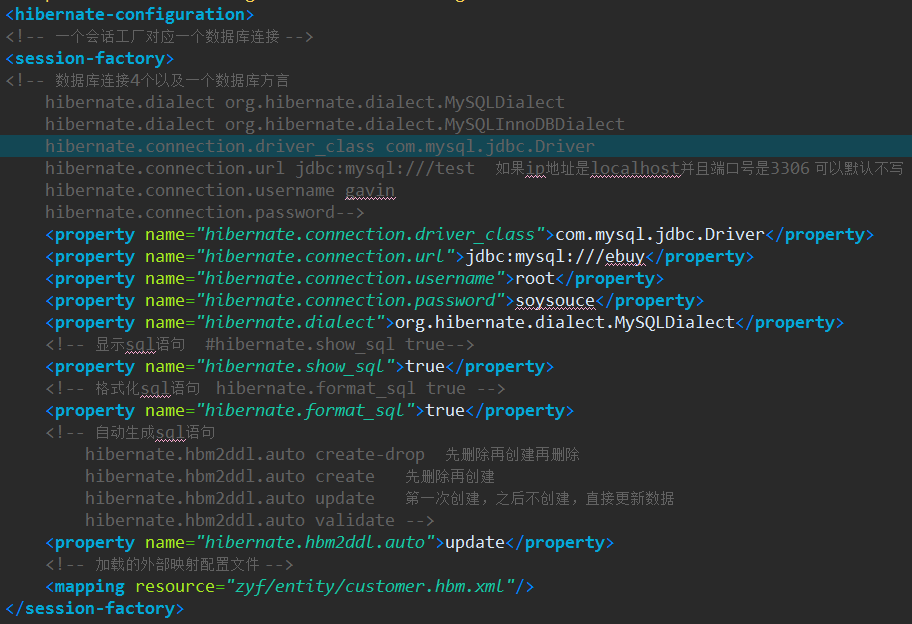






核心配置文件的文件名必须是*hibernate.cfg.xml*  ,xml里面的文件头在核心jar包的文件中复制10-12行（上面是核心配置文件的头，下面是映射文件的头）

格式：

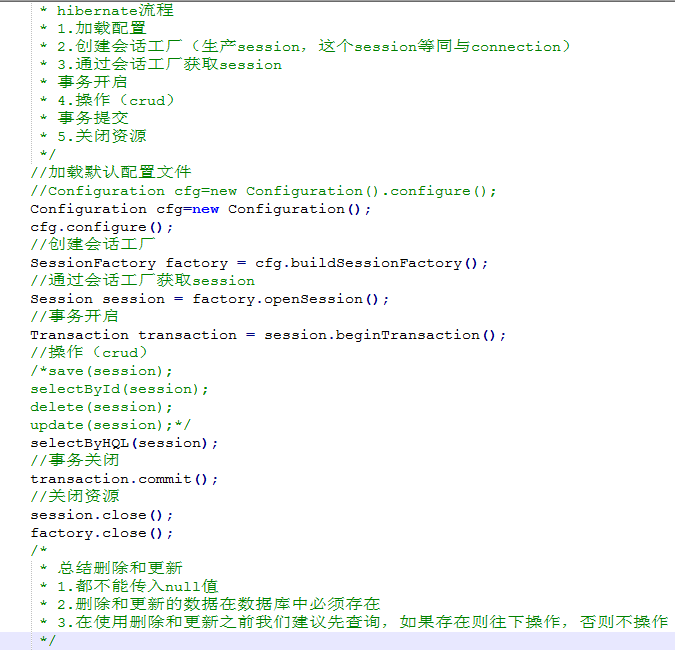


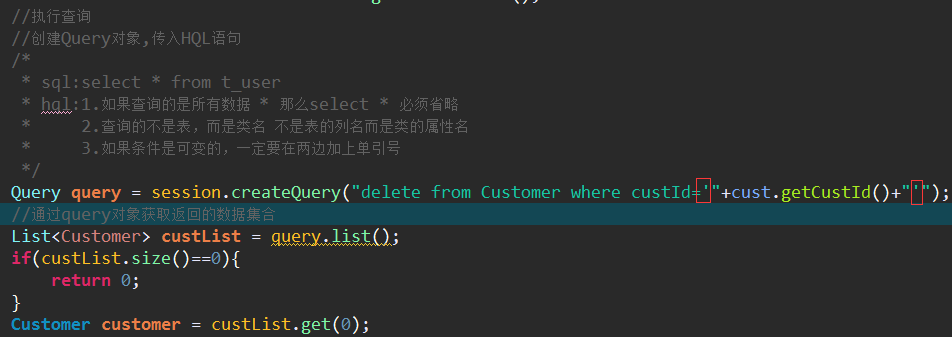
**二、映射文件**





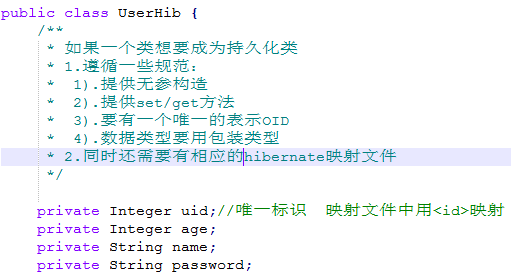
**三、访问方式**



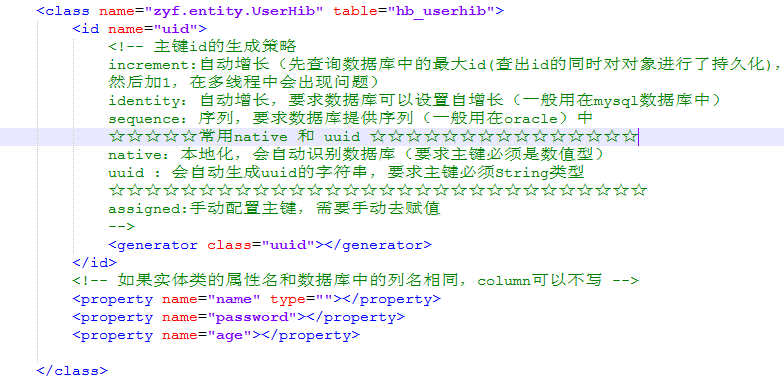




1. **持久化类**



1. **ID生成策略**



1. **持久化对象**

通过持久化类去创建出来的对象，并且该对象的属性值在数据库中有对应的记录存在

持久化对象有4状态：

1. 瞬时状态：通过new关键字创建的对象

session.save(obj):可以将对象从瞬时状态转变为持久状态

1. 持久状态：在session的管理中，并且在数据库张有对应的OID(OID可理解为存在数据，即在数据库中有唯一的标识)

Session的缓存管理(3个方法)：

1. evict(Object obj):去除指定对象obj
2. Clear():清空session缓存
3. Close():关闭session

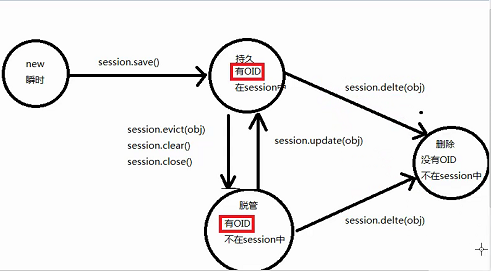
这三个方法都可以将对象从持久状态转变为托管状态（3个方法 都是由session调用）

1. 托管（游离）状态：不在session的管理中，但在数据库中也有对应的OID

session.update(obj)/session.get(obj):可以将对象从托管状态转变为持久状态

1. 删除状态：不在session的管理中，在数据库中也没有对应的OID

Session.delete(obj):可以将对象从托管状态/持久状态转变为删除状态（delete方法先将数据从数据库中删除，删除之后没有了OID，所以session不管理该对象）

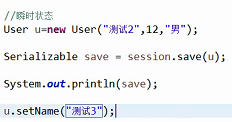


特点：持久化对象如果在持久状态，session会自动更新他的数据

结论：调用delete方法后变成删除状态

调用evict(obj)、clear()、close()变成托管状态

其他就是持久状态

在数据库中uname的值为“测试3”

在数据库中没有uname的值为“测试4”的记录

1. **一级缓存**
2. 为什么需要缓存？

答：我们将一些常用的数据存放在内存中，减少数据库的连接，提高访问速度

Session缓存的快照（相当于备份缓存区的数据）

1. **Session的线程绑定**

事务：原子性

一致性

隔离性

持久性

事务的隔离级别：

不提交读 1

提交读 2（oracle默认级别）

可重复度 4（mysql默认级别）

序列化读 8

1. 为什么将session绑定到线程中？

答：让我们可以在同一个线程的任何地方访问到的session都是同一个对象

如何操作：

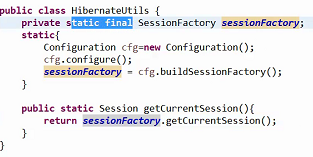
1. 配置主配置文件hibernate.cfg.xml



<!-- 把session绑定到当前线程 -->

<**property**name=*"hibernate.current\_session\_context\_class"*>thread</**property**>

1. 在代码中用Session **session** =factory.getCurrentSession();的方法获取当前线程的session对象（session对象必须是由同一个SessionFactory创建）



重点：hibernate需要手动提交事务（commit），默认情况下不会自动提交事务

openSession()和getCurrentSession()的区别：

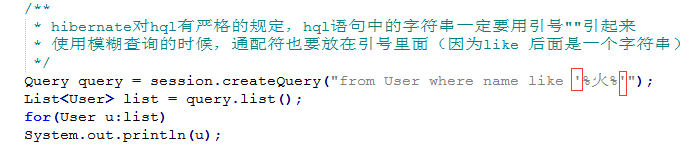
1.如果使用的是openSession()方法，需要调用close()方法手动关闭session

2.如果使用的是getCurrentSession()方法，不需要调用close()方法手动关闭session，在commit或rollback的时候会自动关闭session

1. **查询接口**

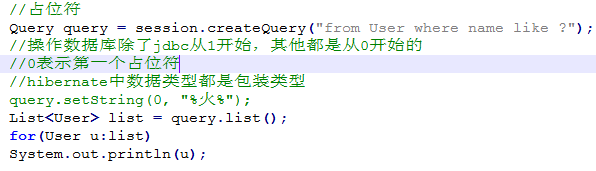
Hql语句：查询语句

1. 普通查询语句

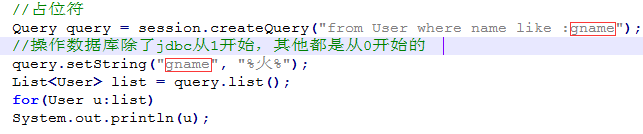


1. 替换占位符（有两种方式）

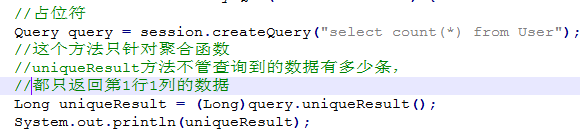
第一种：



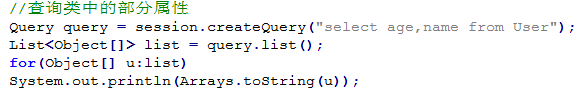
第二种（这种方法是hibernate特有的）：这种方法的好处是可以做到见名知意



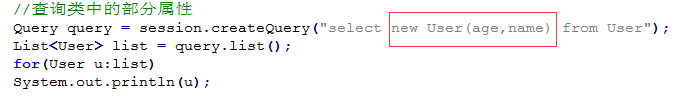
1. 针对聚合函数的查询方式（唯一结果集）



1. 只查询类中的部分属性（两种方式）



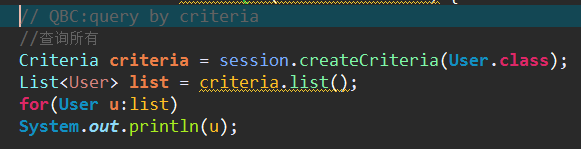
下面这种方法的前提是在User实体类中必须有这个构造方法



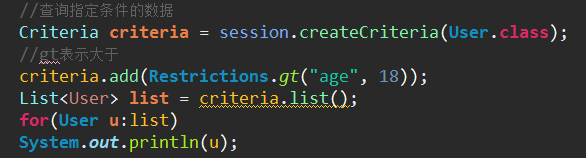
1. **Hibernate高级**

QBC查询

1. 查询全部

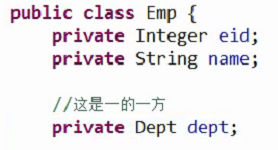


1. 条件查询



1. **关联映射**
2. **一对多**

实体类规范：



一：一个员工只属于一个部门（从当前类的角度出发）



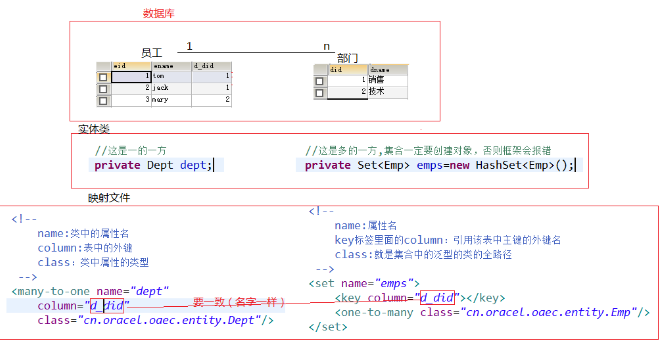
多：一个部门可以有多个员工

配置文件：





对比图：



cascade级联操作:

默认none： 没有

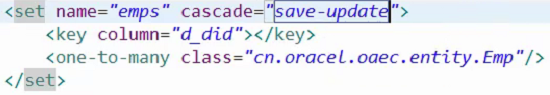
save-update： 保存或更新

delete: 删除

all： 保存、更新、删除

delete-orpan： 孤儿删除

all-delete-orphan： 保存、更新、删除、孤儿删除



在部门（多方）设置级联操作，所以保存部门时，与部门关联的所有员工都会被保存（系统会自动保存员工）

Inverse ：外键维护设置

默认情况下都是false,就是维护外键

true:放弃维护外键（放弃不管）

cascade:做保存、更新、修改时有用

inverse:与外键有关

**在一的一方配置cascade属性<many-to-one>，（保存对象的时候**

**保存一的一方的对象，原因：执行的sql语句更少 ----> 需要保存什么对象，就在该对象对应的映射文件中设置cascade。**

**如果保存“一方”对象并设置cascade，这样在保存”一方”的时候会先创建“多方”的表（在一级缓存中）并插入相应数据，然后创建“一方”的表并执行插入语句插入相应数据，不必执行update语句更新“一方”的外键，这样执行的sql语句就会更少，外键自然就可以只由“一方”维护；**

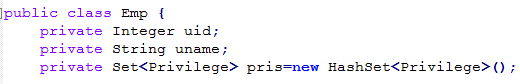
**如果保存“多方”的对象，并在“多方”配置文件中设置cascade，这样在保存对象时，会先创建“多方”和“一方”的表到数据库并执行相应的insert语句插入数据，然后检查cascade级联，如果设置级联操作就会到数据库中对“一方”表执行更新语句设置其外键，这样除了执行插入语句之外，还要执行更新语句，执行的sql语句更多了，效率会变低 所以建议在“一方”的<many-to-one>标签中设置cascade级联操作）**

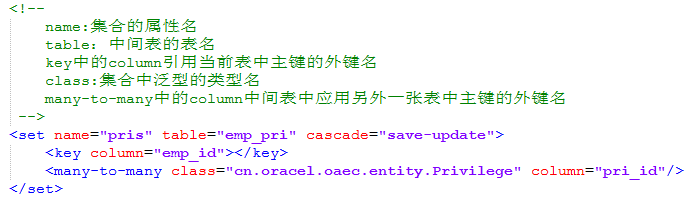
**在多的一方配置inverse=”true”属性<set>**

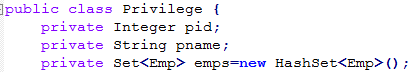
**总结：一的一方 维护外键，级联操作（级联两方都可以设置，建 议设置一的一方）**

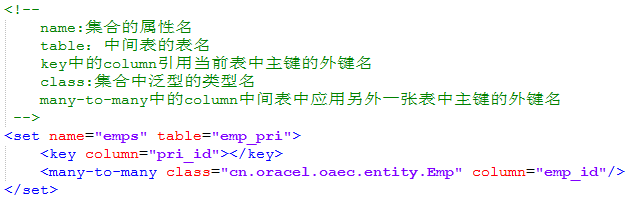
**多的一方 不维护不级联**

1. **多对多**

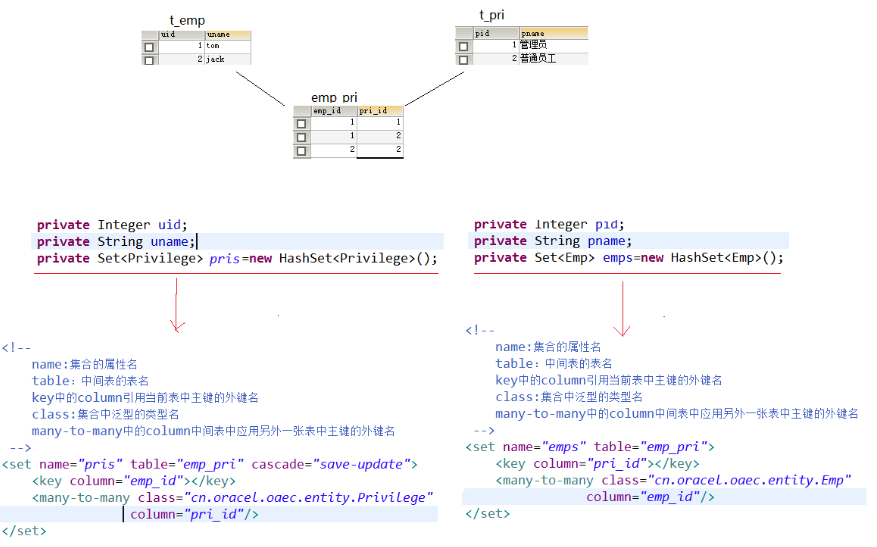








对比图：



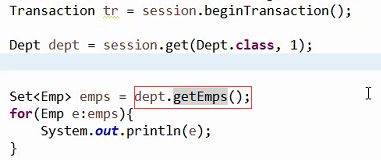
1. **Hibernate查询**
2. **唯一标识列OID查询**



上面的get方法中1就是OID，可以理解为id

Hibernate中一定要开启事务，但是查可以不用提交事务

1. **对象导航查询**



查询的是部门，但是可以得到所有的员工，不用自己调用get方法到数据库查询

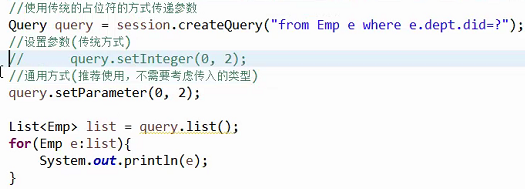
1. **HQL查询**

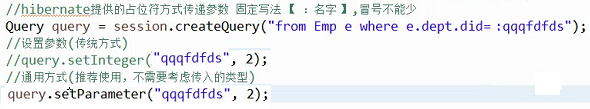
**1）查询全部**



**2）条件查询**

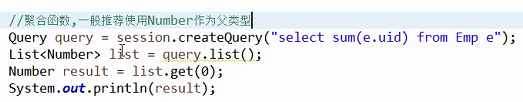






? 和 : 两种方式都可以使用，推荐使用 : 的方式 ，但是？ 和 ：两种方式不要交替使用

**3）聚合函数**



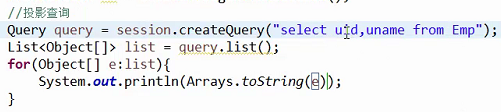


到Number对象中只有数据，没有类型，可以用intValue()等方法得到自己想要的数据类型

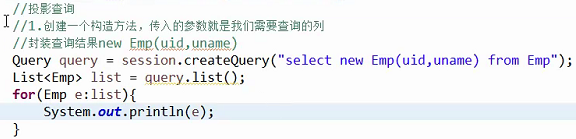
**4）排序查询：**

降序：desc

**5）投影查询**



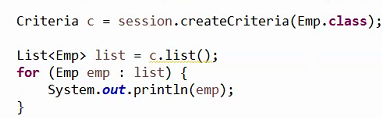
如果要将数组封装到对象中，则要在实体类中创建构造方法（传入参数为需要查询的列（语句中要写类的属性）），封装查询结果



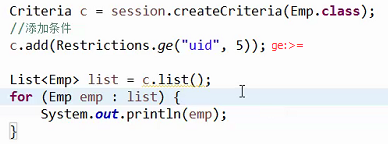
**6）分页查询**



1. **QBC查询**
2. **普通查询**



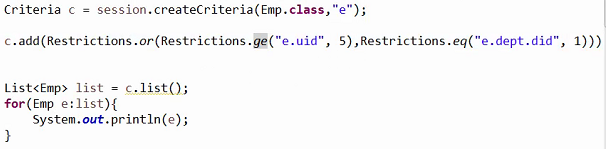
1. **带条件查询**



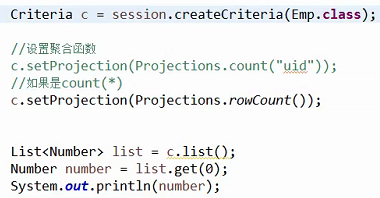
and:



or：

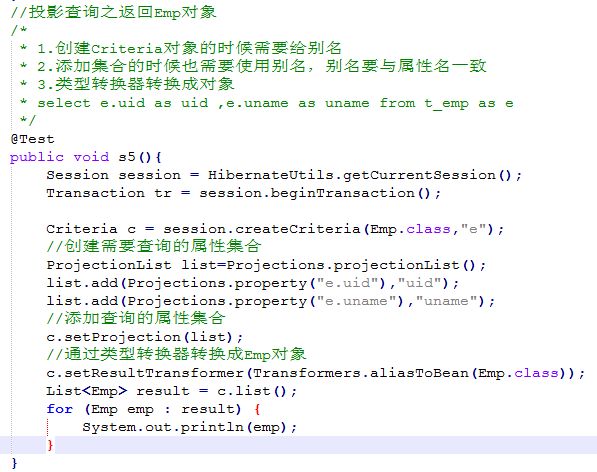


**3）聚合函数查询**

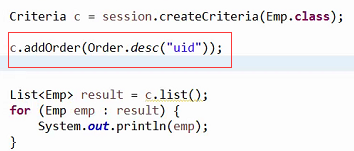


**4）投影查询**

（返回object数组）

（返回对象）

**5）排序查询**



**6）分页查询**



1. **SQL查询**
2. 普通查询

