目录

[Hibernate概述 11](#_Toc17346)

[什么是框架 11](#_Toc14524)

[什么是hibernate 11](#_Toc30659)

[什么是ORM思想 11](#_Toc1959)

[Hibernate入门 12](#_Toc17410)

[搭建hibernate环境 12](#_Toc4618)

[第一步：导入hibernate的jar包 12](#_Toc2332)

[第二步：创建实体类 13](#_Toc284)

[第三步：配置实体类和数据库表一一对应（映射关系） 13](#_Toc16230)

[（1） 创建xml格式配置文件 13](#_Toc6101)

[（2） 配置是xml格式，先引入约束条件 14](#_Toc530)

[（3） 配置映射关系 14](#_Toc8301)

[第四步 创建hibernate的核心配置文件 14](#_Toc18199)

[（1） 核心配置配置文件格式xml,但是核心配置文件名称和位置固定的 14](#_Toc17175)

[（2） 引入DTD约束 15](#_Toc24463)

[第一步：配置数据库信息 15](#_Toc30575)

[第二步：配置hibernate信息 15](#_Toc19429)

[第三步：把映射文件放到核心配置文件中 16](#_Toc3695)

[实现添加操作 16](#_Toc2690)

[第一步 加载hibernate核心配置文件 16](#_Toc227)

[第二步 创建SessionFactory对象 16](#_Toc31969)

[第三步 使用SessionFactory创建Session对象 17](#_Toc20153)

[第四步 开启事务 17](#_Toc7720)

[第五步 写具体逻辑crud操作 17](#_Toc23973)

[第六步 提交事务 17](#_Toc24514)

[第七步 关闭资源 17](#_Toc31202)

[Hibernate配置文件详解 18](#_Toc25483)

[Hibernate映射配置文件 18](#_Toc29222)

[Hibernate核心配置文件 18](#_Toc18567)

[Hibernate核心API 19](#_Toc20474)

[Configuration 19](#_Toc9800)

[SessionFactory（重点） 19](#_Toc30747)

[（1） 创建对象过程中会做以下事情 19](#_Toc29790)

[（1）写工具类，写静态代码块（静态代码块在类加载时执行一次） 19](#_Toc6795)

[Session（重点） 20](#_Toc18068)

[Transaction 21](#_Toc22647)

[实体类编写规则 21](#_Toc19397)

[1. 实体类的属性是private的 21](#_Toc30250)

[2. 私有属性要使用public的get/set方法 21](#_Toc11393)

[3. 要求实体类有属性作为唯一值（一般使用id值） 21](#_Toc24983)

[4. 实体类属性建议不使用基本数据类型，使用基本数据类型对应的包装类 21](#_Toc4079)

[5. 比如表示学生的分数 int Score; 22](#_Toc23115)

[Hibernate主键生成策略 22](#_Toc126)

[（1） 使用uuid生成策略，实体id属性类型必须是String 23](#_Toc11709)

[（2） 配置部分写出uuid值 23](#_Toc30569)

[实体类操作 24](#_Toc21338)

[对实体类crud操作 24](#_Toc19741)

[添加操作 24](#_Toc19904)

[根据ID查询 24](#_Toc32253)

[修改操作 25](#_Toc8967)

[1. 先查询，再修改 25](#_Toc28186)

[（1） 根据ID查询，返回对象 25](#_Toc3074)

[（2） 用对象的Set方法进行设置 25](#_Toc27607)

[（3） 调用session的update方法实现修改 25](#_Toc15073)

[删除操作 25](#_Toc28824)

[实体类对象状态（概念） 26](#_Toc23457)

[实体类状态有三种（没有跟着Session的语句就是与session没关系） 26](#_Toc22502)

[（1） 瞬时态：对象没有id值，对象与session没有关联 26](#_Toc17702)

[（2） 持久态：对象有id值，对象与session有关联 26](#_Toc18605)

[（3） 托管态：对象有id值，对象与session无关联 26](#_Toc18418)

[演示操作实体类对象的方法 26](#_Toc10629)

[（1） saveOrUpdate方法 26](#_Toc12249)

[Hibernate的一级缓存 27](#_Toc33)

[什么是缓存 27](#_Toc30763)

[Hibernate缓存 27](#_Toc23899)

[验证一级缓存存在 28](#_Toc28182)

[Hibernate一级缓存执行过程 28](#_Toc17011)

[Hibernate一级缓存特性 29](#_Toc7007)

[Hibernate事务相关操作 29](#_Toc4479)

[事务概念 29](#_Toc25507)

[Hibernate事务代码规范写法 30](#_Toc27165)

[Hibernate绑定session（实现单线程） 31](#_Toc20656)

[（1） 在hibernate核心配置文件中配置 31](#_Toc7392)

[（2） 在工具类里调用sessionFactory里面的方法得到 31](#_Toc11975)

[（3） 不用手动关闭session(session.close();)线程结束自动关闭 31](#_Toc29465)

[Hibernate的API使用 32](#_Toc24512)

[Query对象 32](#_Toc13797)

[Criteria对象 32](#_Toc17503)

[SQLQuery对象 33](#_Toc30372)

[1.以对象形式返回（推荐） 33](#_Toc21864)

[2. 以数组形式返回 33](#_Toc27728)

[表与表之间关系回顾 34](#_Toc4654)

[Hibernate的一对多操作 35](#_Toc5186)

[一对多映射配置 35](#_Toc22701)

[第一步 建立两个实体类 客户 和 联系人 35](#_Toc31336)

[第二步 让两个实体类之间相互表示 35](#_Toc7029)

[（1） 在客户实体类里面表示多个联系人 35](#_Toc27650)

[（2） 在联系人实体类表示所属客户 36](#_Toc19439)

[第三步 配置映射关系 36](#_Toc12631)

[（1） 一般一个实体类对应一个配置文件 36](#_Toc17347)

[（2） 把映射最基本配置完成 36](#_Toc3587)

[（3） 在映射文件中，配置一对多关系 36](#_Toc3684)

[第四步 创建核心配置文件，把映射文件引入到核心配置文件中 37](#_Toc1233)

[一对多级联操作 37](#_Toc15375)

[一对多级联保存 37](#_Toc823)

[第一步 在客户映射文件中进行配置 38](#_Toc29125)

[第二步 创建客户和联系人对象，只需要把联系人放到客户里面就可以了，最终只需save客户 38](#_Toc31222)

[一对多级联删除 39](#_Toc17924)

[1. 删除某个客户，把客户里面的联系人都删除 39](#_Toc21351)

[2. 具体实现 39](#_Toc14407)

[第一步 在客户映射文件Set标签进行配置 39](#_Toc32122)

[（1） 使用属性cascaed属性值delete 39](#_Toc16796)

[第二步 在代码中直接删除客户 39](#_Toc11384)

[（1） 根据ID查询对象，调用session的delete方法 39](#_Toc11981)

[3.执行过程 39](#_Toc25859)

[（1）根据ID查客户 39](#_Toc14576)

[（2）根据外键ID查询所有联系人 39](#_Toc3820)

[（3）把联系人外间设置成null 39](#_Toc14172)

[（4）删除客户和联系人 39](#_Toc8146)

[一对多修改操作（inverse属性） 39](#_Toc24143)

[1. 让Tom联系人由Yoon改成Piage 39](#_Toc9132)

[2.inverse属性 40](#_Toc22479)

[Hibernate多对多操作 40](#_Toc23206)

[多对多映射配置 40](#_Toc12429)

[第一步 创建实体类 40](#_Toc7940)

[第二步 让两个实体类之间相互表示 40](#_Toc8396)

[（1） 用户里面表示所有角色，使用set集合 40](#_Toc6877)

[（2） 一个角色里面有多个用户，使用set集合 41](#_Toc14749)

[第三步 配置映射关系 41](#_Toc21743)

[（1） 基本配置 41](#_Toc246)

[（2） 配置多对多关系 41](#_Toc5341)

[第四步 在核心配置文件中引入映射文件 42](#_Toc7010)

[多对多级联保存 42](#_Toc25237)

[第一步 在用户配置文件中set标签进行配置，cascade值save-update 42](#_Toc22628)

[第二步 写代码实现 42](#_Toc16818)

[（1）创建用户和角色对象，把角色放到用户里面，保存用户 42](#_Toc24259)

[多对多级联删除（了解，一般不用） 42](#_Toc4370)

[第一步 在set标签设置cascade值delete 42](#_Toc24659)

[第二步 调用session方法delete 42](#_Toc30712)

[维护第三张表关系（重点） 42](#_Toc14590)

[1. 用户和角色多对多关系，维护关系通过第三张表维护 42](#_Toc21881)

[2. 让某个用户有某个角色 42](#_Toc15127)

[第一步 根据ID查询用户和角色 42](#_Toc3068)

[第二步 把角色放到用户set集合里面 42](#_Toc15166)

[3. 让某个用户删除某个角色 43](#_Toc4632)

[第一步 根据ID查询用户和角色 43](#_Toc16472)

[第二步 把角色从用户set集合移除 43](#_Toc15134)

[Hibernate查询方式 43](#_Toc8303)

[1. 对象导航查询 43](#_Toc31998)

[2. OID查询 44](#_Toc22906)

[3. hql查询 44](#_Toc26234)

[（1）HQL:hibernate query language 44](#_Toc11601)

[（2）常用的HQL语句 44](#_Toc20411)

[（3）使用HQL语句时，要使用Query对象 44](#_Toc605)

[1.创建Query对象，写hql语句 44](#_Toc24775)

[2.调用query对象里面的方法得到结果 44](#_Toc22219)

[查询所有 45](#_Toc9319)

[1. 查询所有客户的记录 45](#_Toc16967)

[（1）创建Query对象，写hql语句 45](#_Toc21374)

[（2）调用query对象里面的方法得到结果 45](#_Toc8835)

[2. 查询所有，from 实体 类名称 45](#_Toc9190)

[条件查询 45](#_Toc12574)

[1. HQL条件查询语句写法 45](#_Toc32575)

[2.代码 45](#_Toc30123)

[排序查询 46](#_Toc26985)

[1. hql排序语句 46](#_Toc23771)

[分页查询 46](#_Toc23410)

[1. mysql实现分页：使用关键字LIMIT实现 46](#_Toc17950)

[2. 在hql实现分页查询 46](#_Toc16184)

[投影查询 47](#_Toc9019)

[1. 查询部分字段（E.g Select custId from Customer） 47](#_Toc22826)

[2. 投影查询HQL语句写法 47](#_Toc25003)

[聚集函数使用 47](#_Toc6320)

[（1） 常用的聚集函数 47](#_Toc21431)

[（2） hql聚集函数语句写法 47](#_Toc19790)

[4. QBC查询 48](#_Toc16304)

[1. 使用HQL查询需要些HQL语句，使用QBC查询调用方法即可 48](#_Toc28039)

[2. 使用QBC时，使用实体类和属性 48](#_Toc11703)

[3. 使用QBC，使用Criteria对象 48](#_Toc17914)

[查询所有 48](#_Toc4331)

[条件查询 48](#_Toc23128)

[模糊查询 49](#_Toc3843)

[排序查询 49](#_Toc20259)

[分页查询 50](#_Toc398)

[统计查询 50](#_Toc10843)

[离线查询（不用Session.creatCriteria） 50](#_Toc27258)

[HQL多表查询 51](#_Toc24977)

[MySQL多表查询 51](#_Toc11891)

[1.内连接 51](#_Toc16134)

[2.左外连接 51](#_Toc16019)

[3.右外连接 51](#_Toc10060)

[HQL实现多表查询 51](#_Toc1391)

[内连接 51](#_Toc30415)

[（1） 内连接hql语句写法，以客户和联系人为例 51](#_Toc16711)

[迫切内连接 52](#_Toc69)

[左外连接 53](#_Toc14729)

[迫切左外连接 53](#_Toc30383)

[右外连接 53](#_Toc24919)

[Hibernate检索策略 53](#_Toc25671)

[1. hibernate检索策略分为两类 53](#_Toc22323)

[E.g：只要对象的id，用load方法效率比get方法高很多 54](#_Toc32231)

[2. 延迟查询分成两类 54](#_Toc29521)

[（1） 类级别查询：根绝ID查询返回实体类对象，调用load方法不会立即发送语句 54](#_Toc24244)

[（2） 关联级别查询： 54](#_Toc24852)

[关联级别延迟操作 54](#_Toc22032)

[1.在映射文件中进行配置 54](#_Toc26040)

[（1）根据客户得到所有联系人，在客户配置文件中配置 54](#_Toc539)

[2.在set标签上使用属性 54](#_Toc17741)

[（1）fetch：值select 54](#_Toc852)

[（2）lazy：值 54](#_Toc95)

[批量抓取 55](#_Toc14861)

# Hibernate概述

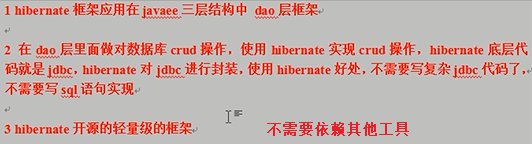
WEB应用开发sessionFactory不需要关闭

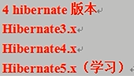
**（MySQL5.7以上方言要设置成org.hibernate.dialect.MySQL5InnoDBDialect）**

## 什么是框架



## 什么是hibernate

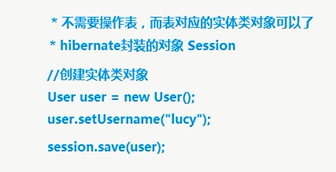
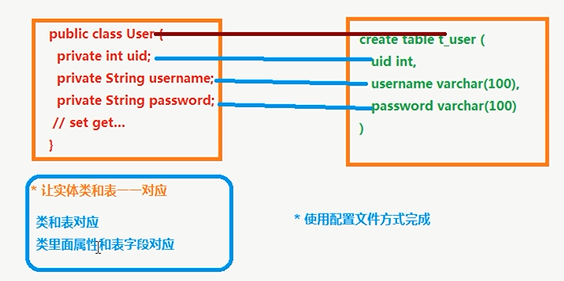




## 什么是ORM思想



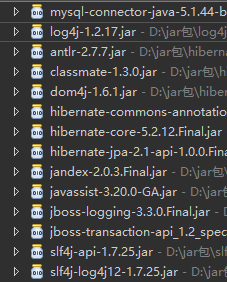
画图描述：



# Hibernate入门

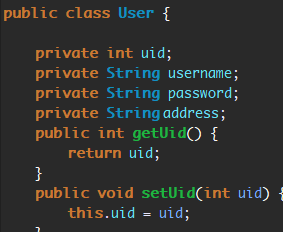
## 搭建hibernate环境

**第一步：导入hibernate的jar包**



因为使用Hibernate时有日志信息输出，hibernate本身没有日志输出的jar包，导入其他的日志jar包 **不要忘了mysql的jar包！！！**

**第二步：创建实体类**



使用Hibernate时不需要自己创建表，会帮我们创建

**第三步：配置实体类和数据库表一一对应（映射关系）**

使用配置文件实现映射关系

1. 创建xml格式配置文件

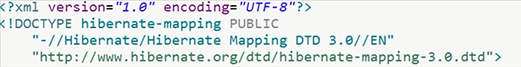
-映射配置文件名称和位置没有固定要求

-**建议**在**实体类包里面创建**，名称用实体类名.hbm.xml

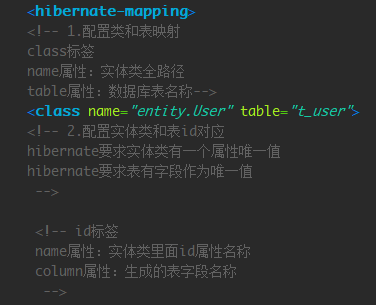
1. 配置是xml格式，先引入约束条件

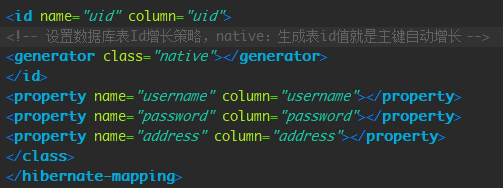
-学过约束dtd,schema，在hibernate文件中引入dtd约束





1. 配置映射关系



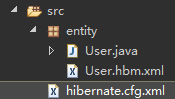


**第四步 创建hibernate的核心配置文件**

1. 核心配置配置文件格式xml,但是核心配置文件名称和位置固定的

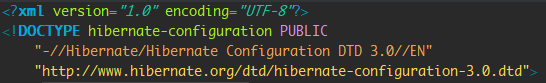
-位置：必须src下面

-名称：必须hibernate.cfg.xml



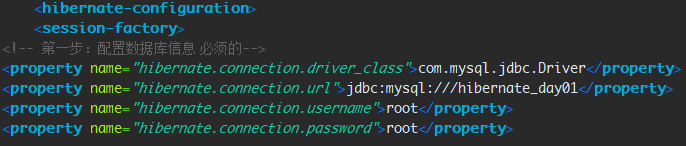
1. 引入DTD约束



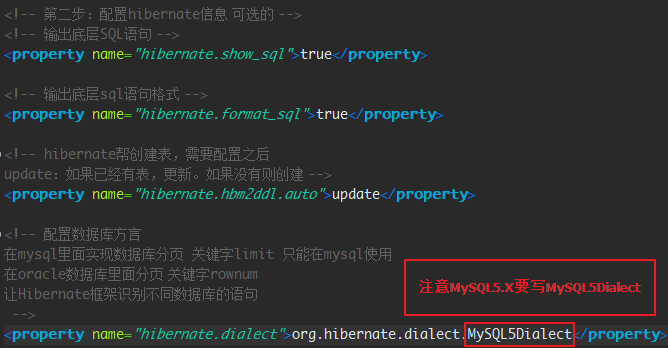


hibernate操作过程中，只会加载核心配置文件，其他配置文件不会加载

第一步：配置数据库信息

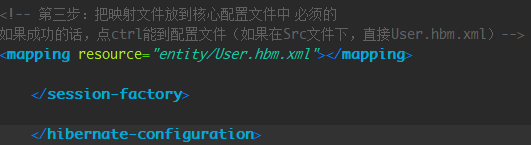


第二步：配置hibernate信息



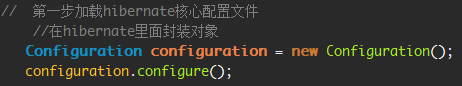
**（MySQL5.7以上方言要设置成org.hibernate.dialect.MySQL5InnoDBDialect）**

第三步：把映射文件放到核心配置文件中

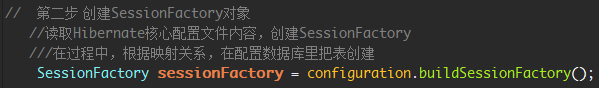


## 实现添加操作

第一步 加载hibernate核心配置文件



第二步 创建SessionFactory对象



第三步 使用SessionFactory创建Session对象



第四步 开启事务



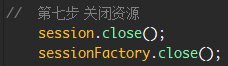
第五步 写具体逻辑crud操作



第六步 提交事务



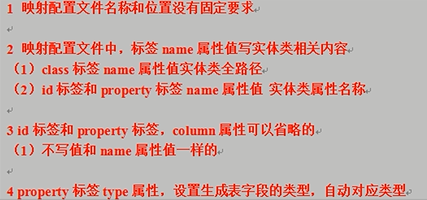
第七步 关闭资源



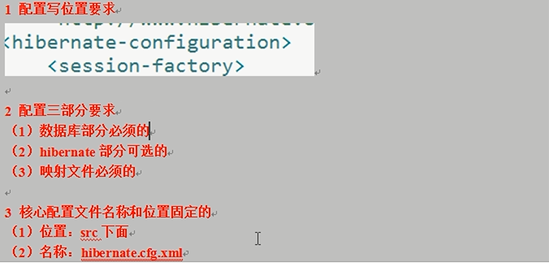
**只有第五步是变化的，其他都相同**

# Hibernate配置文件详解

## Hibernate映射配置文件

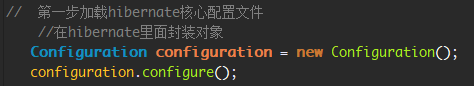


## Hibernate核心配置文件



# Hibernate核心API

## Configuration



到src下面找到名称hibernate.cfg.xml配合文件，创建对象，把配置文件放到对象里面（加载核心配置文件）

## SessionFactory（重点）

1. 使用Configuration对象创建SessionFactory对象
2. 创建对象过程中会做以下事情

-根据核心配置文件中，有数据库配置，有映射文件部分，到数据库里面根据映射关系把表创建

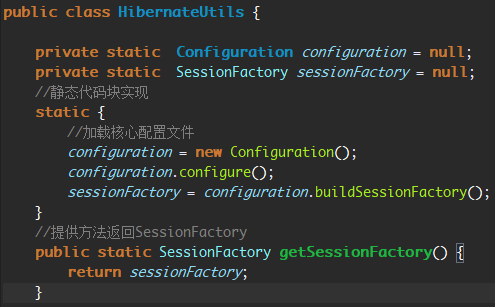


自动创建表语句

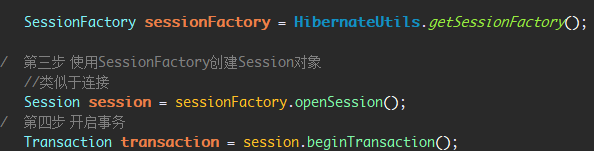
2.创建SessionFactory过程非常耗资源，so，在一个项目中，一般只创建一个sessionFactory的对象

3.实现方法

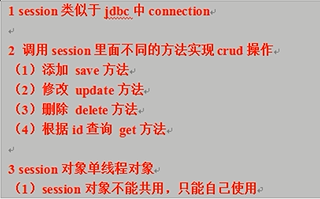
（1）写工具类，写静态代码块（静态代码块在类加载时执行一次）

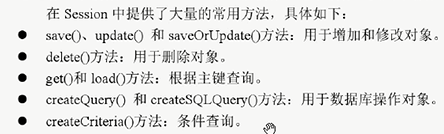


测试：

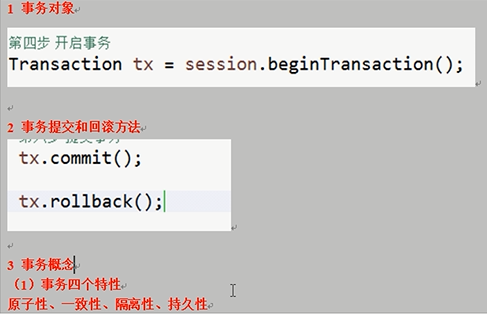


## Session（重点）





## Transaction



# 实体类编写规则

1. 实体类的属性是private的
2. 私有属性要使用public的get/set方法
3. **要求实体类有属性作为唯一值（一般使用id值）**
4. **实体类属性建议不使用基本数据类型，使用基本数据类型对应的包装类**

**-八个数据基本类型对应的包装类**

**Int-Integer**

**Char-Character**

**其他的都是首字母大写**

1. 比如表示学生的分数 int Score;

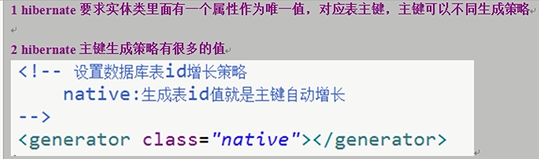
-比如学生得0分，score=0

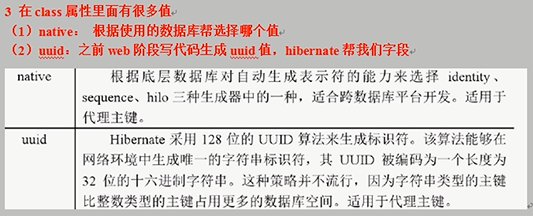
-如果表示学生没有参加考试，int score=0不能准确表示学生是否参加考试

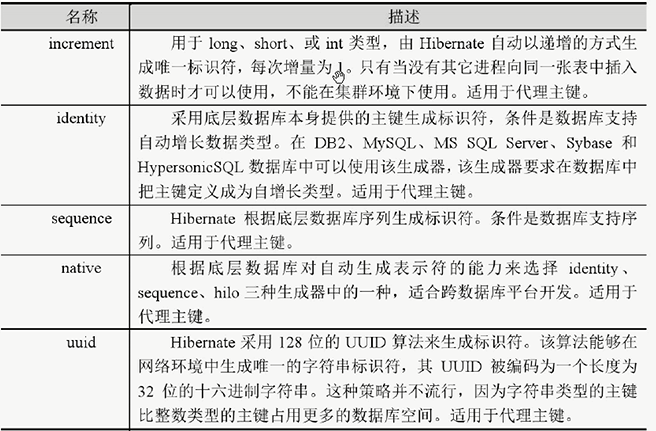
**\*解决：使用包装类 Integer Score = 0;表示学生得0分**

**Integer Score = null;表示学生没有参加考试**

# Hibernate主键生成策略

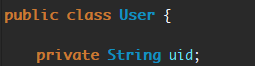






uuid生成策略演示：

1. 使用uuid生成策略，实体id属性类型必须是String



1. 配置部分写出uuid值



# 实体类操作

## 对实体类crud操作

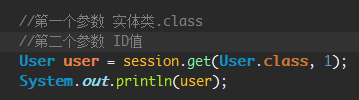
### 添加操作

调用session里面的save方法



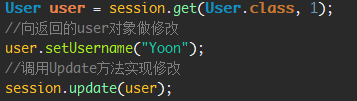
### 根据ID查询

调用session里面的get方法



### 修改操作

1. 先查询，再修改
2. 根据ID查询，返回对象
3. 用对象的Set方法进行设置
4. 调用session的update方法实现修改



**\*用set方法，但是会把没设置的值都传进去，变成Null**

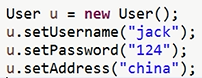
### 删除操作



# 实体类对象状态（概念）

实体类状态有三种（没有跟着Session的语句就是与session没关系）

1. 瞬时态：对象没有id值，对象与session没有关联



（没有跟着session.方法(u)）

1. 持久态：对象有id值，对象与session有关联



（通过session查出来的）

1. 托管态：对象有id值，对象与session无关联



## 演示操作实体类对象的方法

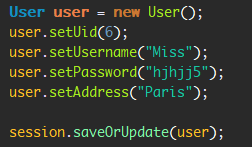
**根据不同状态做不同操作**

1. saveOrUpdate方法

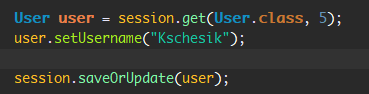
**瞬时态对象（无id,添加操作）**



**托管态操作（有id，修改操作）**



**持久态操作（有id,修改操作）**



# Hibernate的一级缓存

## 什么是缓存



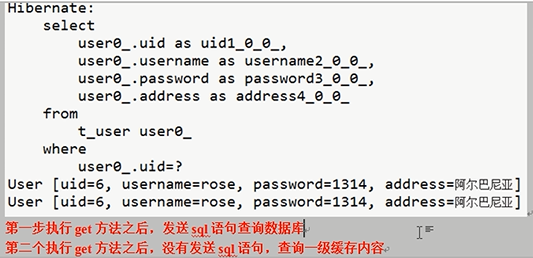
## Hibernate缓存



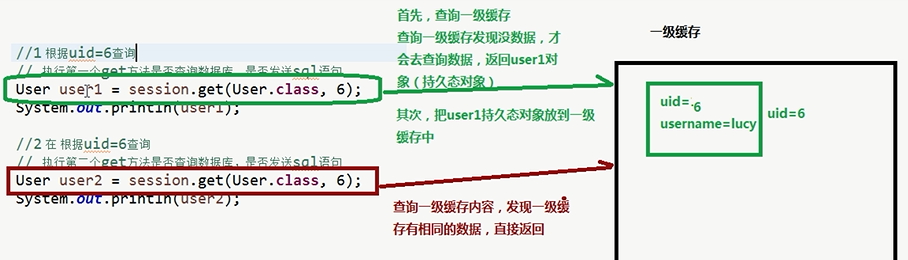
## 验证一级缓存存在





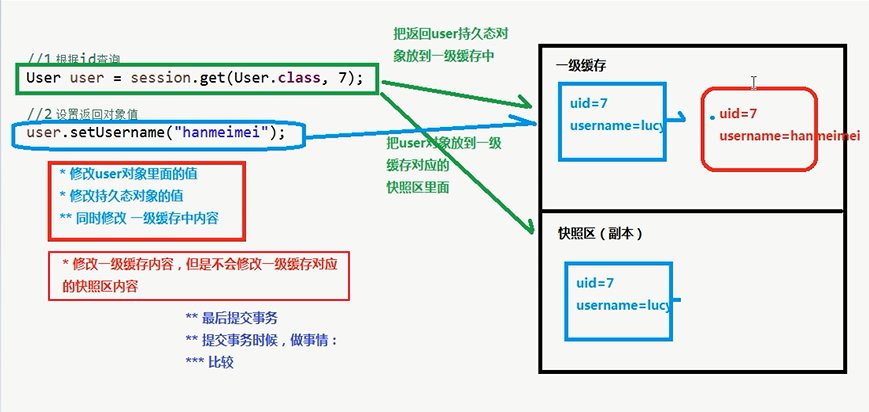


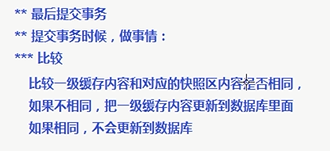
## Hibernate一级缓存执行过程



## Hibernate一级缓存特性

持久太会自动更新数据库





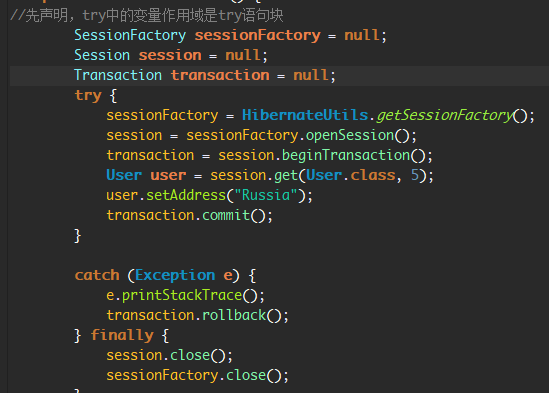
# Hibernate事务相关操作

## 事务概念



## Hibernate事务代码规范写法



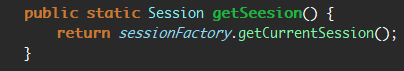


# Hibernate绑定session（实现单线程）

1. session类似于jdbc的connection，之前Web阶段学过threadLocal
2. 帮实现与本地线程绑定session
3. 获取与本地线程session
4. 在hibernate核心配置文件中配置



1. 在工具类里调用sessionFactory里面的方法得到



1. 不用手动关闭session(session.close();)线程结束自动关闭

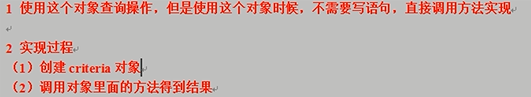
# Hibernate的API使用

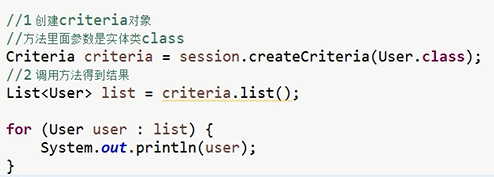
## Query对象



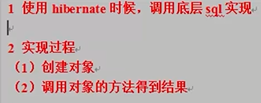


## Criteria对象





## SQLQuery对象



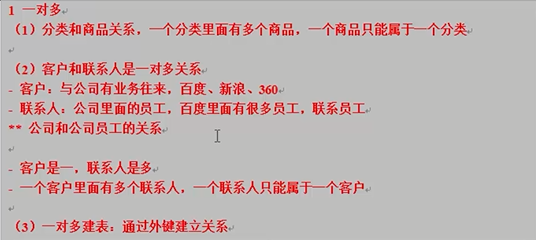
**1.以对象形式返回（推荐）**



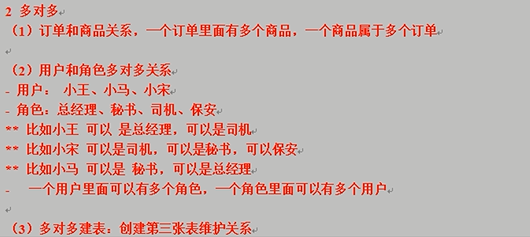
1. **以数组形式返回**



# 表与表之间关系回顾











# Hibernate的一对多操作

## 一对多映射配置

以客户和联系人为例

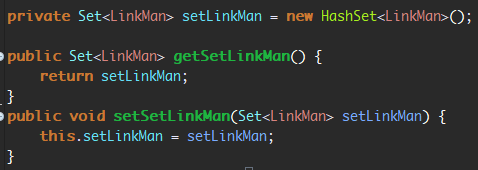
导入jar包

**第一步 建立两个实体类 客户 和 联系人**

**第二步 让两个实体类之间相互表示**

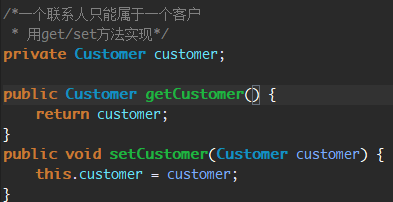
1. 在客户实体类里面表示多个联系人

-一个客户可以有多个联系人



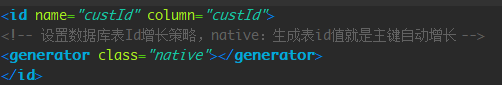
1. 在联系人实体类表示所属客户

-一个联系人只能属于一个客户



**第三步 配置映射关系**

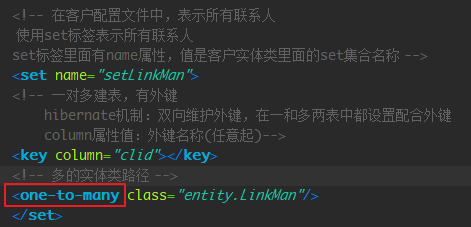
1. 一般一个实体类对应一个配置文件
2. 把映射最基本配置完成



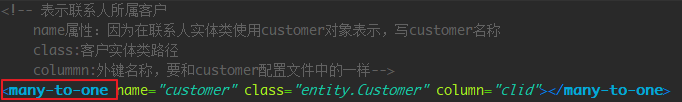
**PS:注意这个custId是实体类里面的一个属性，在下面的property标签里就不用写了**

1. 在映射文件中，配置一对多关系

-在客户配置文件中，表示所有联系人



-在联系人配置文件中，表示所属客户

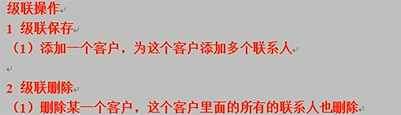


**第四步 创建核心配置文件，把映射文件引入到核心配置文件中**



**（MySQL5.7以上方言要设置成org.hibernate.dialect.MySQL5InnoDBDialect）**

# 一对多级联操作



## 一对多级联保存

**添加客户，为客户添加一个联系人**

复杂方法（clid是自动关联的）





**简化写法**

第一步 在客户映射文件中进行配置

-在客户映射文件set标签中进行配置



第二步 创建客户和联系人对象，只需要把联系人放到客户里面就可以了，最终只需save客户



## 一对多级联删除

1. 删除某个客户，把客户里面的联系人都删除
2. 具体实现

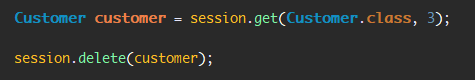
第一步 在客户映射文件Set标签进行配置

1. 使用属性cascaed属性值delete



第二步 在代码中直接删除客户

1. 根据ID查询对象，调用session的delete方法



3.执行过程

（1）根据ID查客户

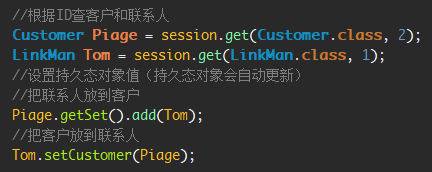
（2）根据外键ID查询所有联系人

（3）把联系人外间设置成null

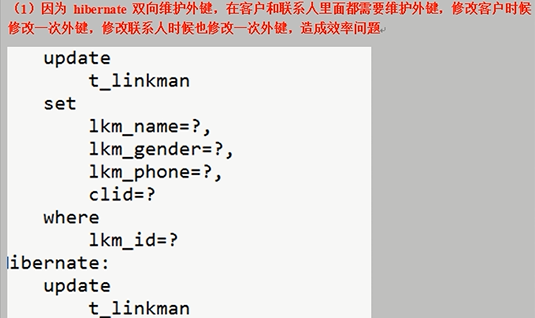
（4）删除客户和联系人

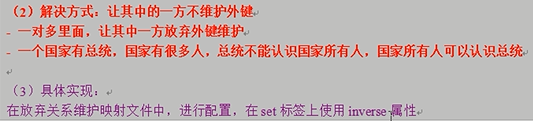
## 一对多修改操作（inverse属性）

1. 让Tom联系人由Yoon改成Piage



2.inverse属性







## Hibernate多对多操作

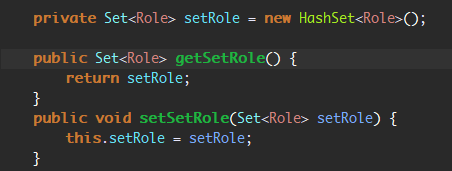
## 多对多映射配置

以用户和角色为例演示

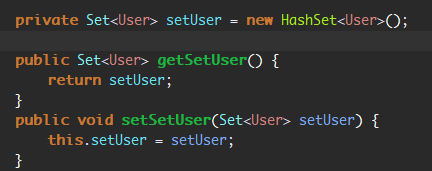
**第一步 创建实体类**

**第二步 让两个实体类之间相互表示**

1. 用户里面表示所有角色，使用set集合



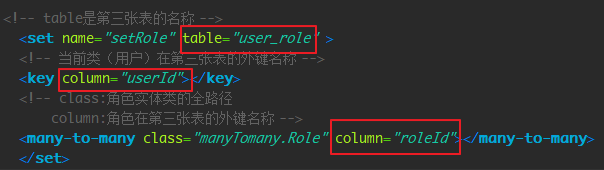
1. 一个角色里面有多个用户，使用set集合



**第三步 配置映射关系**

1. 基本配置
2. 配置多对多关系

**User.hbm.xml**



Role.hbm.xml



**第四步 在核心配置文件中引入映射文件**



## 多对多级联保存

根据用户保存角色

第一步 在用户配置文件中set标签进行配置，cascade值save-update

第二步 写代码实现

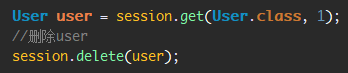
（1）创建用户和角色对象，把角色放到用户里面，保存用户

## 多对多级联删除（了解，一般不用）

把一个用户的角色删除的时候，会把角色删除，连带其他用户的角色也删了

第一步 在set标签设置cascade值delete

第二步 调用session方法delete

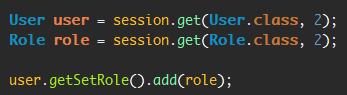


## 维护第三张表关系（重点）

1. **用户和角色多对多关系，维护关系通过第三张表维护**
2. **让某个用户有某个角色**

第一步 根据ID查询用户和角色

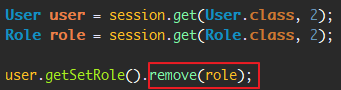
第二步 把角色放到用户set集合里面



1. **让某个用户删除某个角色**

第一步 根据ID查询用户和角色

第二步 把角色从用户set集合移除

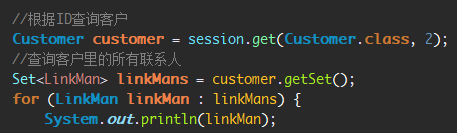


# Hibernate查询方式

## 对象导航查询

根据ID查询某个客户，再查询这个客户里面的所有联系人

代码实现



## OID查询

根据ID查询某一条记录，返回对象



## hql查询

**（1）HQL**:hibernate query language

hibernate提供的一种查询语言，和sql很相似

区别：普通SQL操作数据库表和字段，HQL操作实体类和属性

**（2）常用的HQL语句**

**查询所有 from 实体类名称**

**条件查询 from 实体类名称 where 实体类属性=？ AND 实体类属性=？**

**模糊查询 from 实体类名称 where 实体类属性 like ?**

**排序查询 from 实体类名称 order by 实体类属性名称 [asc（默认升序）/desc（降序）]**

**投影查询 select 属性名称1,属性名称2 from 实体类名称**

**（3）使用HQL语句时，要使用Query对象**

1.创建Query对象，写hql语句

2.调用query对象里面的方法得到结果

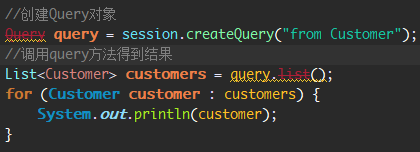
### 查询所有

1. **查询所有客户的记录**

（1）创建Query对象，写hql语句

（2）调用query对象里面的方法得到结果

1. 查询所有，from 实体 类名称



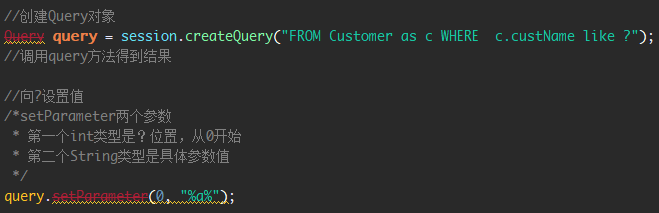
### 条件查询

1. HQL条件查询语句写法
2. **FROM 实体类名称 WHERE 实体类属性名称 = ？ AND 实体类属性名称 = ？**

2.代码



**模糊查询**



### 排序查询

1. hql排序语句

**from 实体类名称 order by 实体类属性名称 [asc（默认升序）/desc（降序）]**

### 

### 分页查询

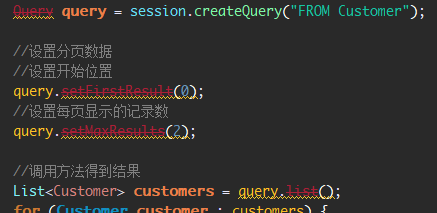
1. mysql实现分页：使用关键字LIMIT实现



（从第0条开始，每页显示3条）

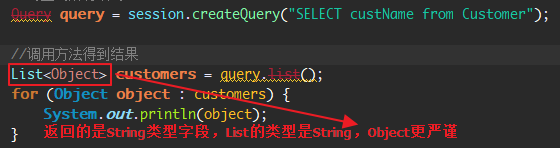
1. 在hql实现分页查询

（1）在hql中，不能写LIMIT，hibernate的query对象中封装了两个方法实现分页操作



### 投影查询

1. 查询部分字段（E.g Select custId from Customer）
2. 投影查询HQL语句写法
3. select 属性名称1,属性名称2 from 实体类名称（select后面不能写\*，不支持）



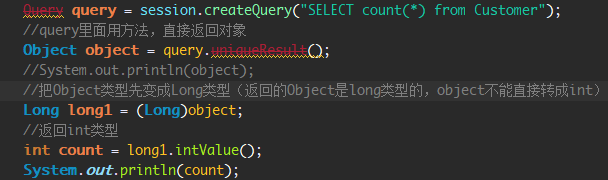
### 聚集函数使用

1. 常用的聚集函数

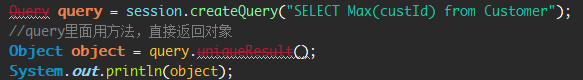
Count avg sum max min

1. hql聚集函数语句写法

Select count(\*) from 实体类名



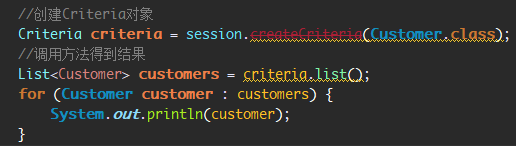
（不用List返回）



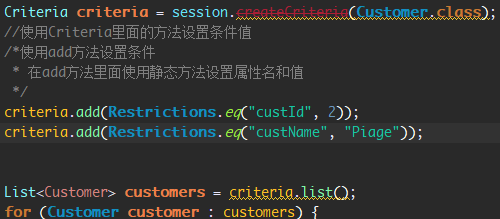
## QBC查询

1. 使用HQL查询需要些HQL语句，使用QBC查询调用方法即可
2. 使用QBC时，使用实体类和属性
3. 使用QBC，使用**Criteria对象**

### 查询所有

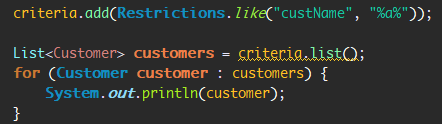


### 条件查询



### 模糊查询

**Restrictions.like**





### 排序查询



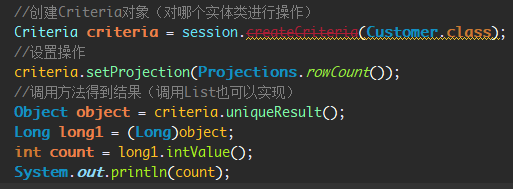
### 分页查询

### 

**开始位置计算公式：（当前页-1）\*每页记录数**

### 统计查询

**统计记录数**



### 离线查询（不用Session.creatCriteria）



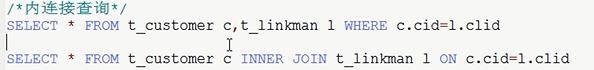




# HQL多表查询

## MySQL多表查询

**1.内连接**



查两个表有关联的数据

**2.左外连接**



左边表所有数据+右边表有关联的数据

**3.右外连接**



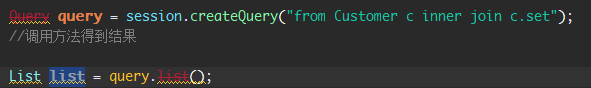
跟左外连接相反

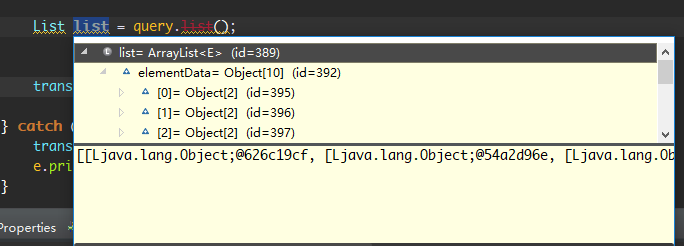
## HQL实现多表查询

## 内连接

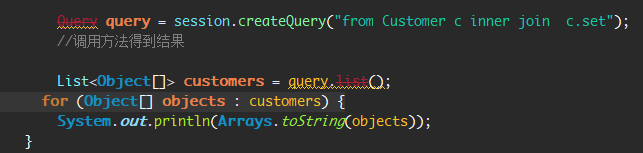
1. 内连接hql语句写法，以客户和联系人为例

**FROM Customer c INNER JOIN c.setLinkMan**





**返回的list里面每部分都是数组**

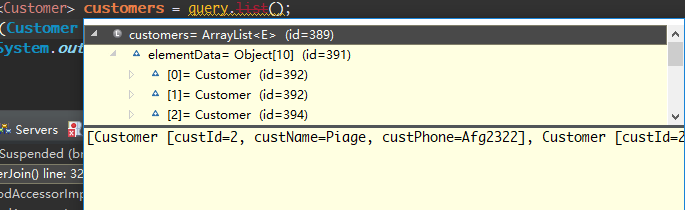


## 迫切内连接

迫切内连接和内连接底层的SQL语句都一样

区别：迫切内连接返回的list中每部分都是对象

Hql语句写法 **FROM Customer c INNER JOIN FETCH c.setLinkMan**





## 左外连接

From Customer c left outer join c.setLinkMan

## 迫切左外连接

From Customer c left outer join fetch c.setLinkMan

区别同内连接和迫切内连接

## 右外连接

From Customer c right outer join c.setLinkMan

# Hibernate检索策略

1. **hibernate检索策略分为两类**
2. 立即查询：根据id查询，调用Get方法，一调用get方法马上发送语句查询数据库



1. 延迟查询：根据id查询，还有load方法，调用Load方法不会马上发送语句查询数据，只有得到对象里面的值才会发送语句查询数据库



**区别：当通过get方法得到实体类对象时（不管要不要用这个对象），会发送语句到数据库查询所有的属性。通过Load方法，只是得到实体类对象的一个代理对象（代理对象里面只有id值），当要操作对象的其他属性时，才会发送语句到数据库。效率更高**

**E.g：只要对象的id，用load方法效率比get方法高很多**

1. **延迟查询分成两类**
2. **类级别查询：根绝ID查询返回实体类对象，调用load方法不会立即发送语句**
3. **关联级别查询：**

**-查询某个用户，再查询这个客户的所有联系人，查询客户的所有联系人的过程是否需要延迟，这个过程称为关联级别延迟**



**（得到linkman的长度才是对属性的操作，hibernate已经优化过）**

## 关联级别延迟操作

1.在映射文件中进行配置

（1）根据客户得到所有联系人，在客户配置文件中配置

2.在set标签上使用属性

（1）fetch：值select

（2）lazy：值

-true：延迟（默认，一般用这个）



-false：立即（一般不用，效率低）



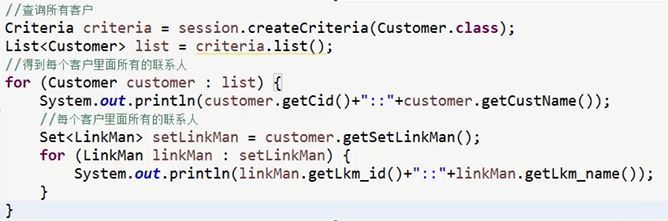
调用get之后，发送两条语句

-extra：极其延迟（最高效）

要什么值给什么值

# 批量抓取





**发送了很多条SQL语句，性能低下**

