将讲解三个点：1、掌握持久类编写（掌握）

2、hibernate主键生成策略（了解）

3、hibernate一级缓存（掌握）

4、hibernate的其他api （掌握）

01-Hibernate的持久化类的编写规则.avi

A：hibernate持久类

持久类就javabean该bean与表建立了映射。

B：hibernate持久类要求

1. 要求有无参构造
2. 类的属性需要私有，对私有的属性提供public的get和set方法。
3. 提供一个唯一标识（OID）的属性与表的主键对应。

Java里面使用地址区分是否是同一个对象。数据库中是通过主键区分是否是同一个记

1. 类不要使用final修饰。如果用final，使用load返回到而是实体对象并不是代理对象。这样就没有用到延迟加载。
2. 类中的属性尽量使用包装类类型。如果没有赋值，基本类型有些是0,会有歧义等，但是包装类型默认是null;

注意：注意持久类书写要求，养成这样的习惯；

02-Hibernate的主键生成策略：主键类型.avi

A：主键的分类

1. 自然主键（业务主键）

该id唯一，有实际意义，如把学生学号作为主键，该主键是有意义的且能够唯一识别每一条学生数据；

1. 代理主键（逻辑主键）

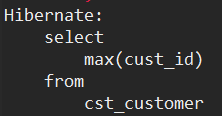
该id唯一，没有实际意义，如不讲学号作为主键，使用自动增长从1开始，每一行数据中包括id(从1开始递增)，学生学号等信息。这里自动递增的id为主键能够识别每一条学生数据；

注意：自然主键该主键是在业务中是有意义的，代理主键该主键在业务中是没有意义的。

03-Hibernate的主键生成策略：演示主键生成策略一.avi

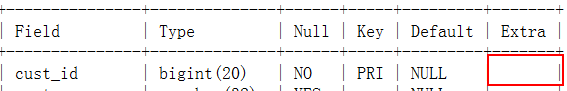
A: 主键生成策略---increment

在customer.hbm.xml文件中<generator class=”increment”></generator>使用increment作为主键的时候。hibernate首先回去数据库查询最大主键的值：



然后再去根据该最大值递增插入数据库；该主键自增需要在没有其他用户向该表插入数据的时候，才能够使用，否则会出现主键重复异常。

通过该主键生成的表并没有在主键后面标注increment



注意：在实际使用hibernate不会使用hibernate提供的increament主键生成策略去生成主键；

04-Hibernate的主键生成策略：演示主键生成策略二.avi

A：主键生成策略---identity

除了让hibernate帮我们产生主键外，我们还可以通过identity来让数据库自己产生主键。前提是该数据库是支持的自动增长即有auto\_increment关键字来标识该主键。目前mysql，ms server是支持auto\_increment,但是oracle数据库是不支持auto\_incrementl;

B：主键生成策略---sequence

该主键采用sequence，前提是该数据库是支持sequence序列的。目前oralce是通过序列来让主键自增的。这里也是通过数据库自己来让主键自增；

C：主键生成策略---native

根据使用数据库种类，自动选择主键生成策略是increment还是sequence.适合跨平台开发，适用于代理主键；

D：主键生成策略---uuid

通过uuid产生的是字符串，所有实体类中主键是String类型的。另外该字符串占据数据库很大空间，在实际开发过程中并不是适用。适合代理主键；

E：主键生成策略---assigned

如果采用该主键生成策略，该必须通过java程序手动给主键set值。这样才能够插入数据库，否则会报错；

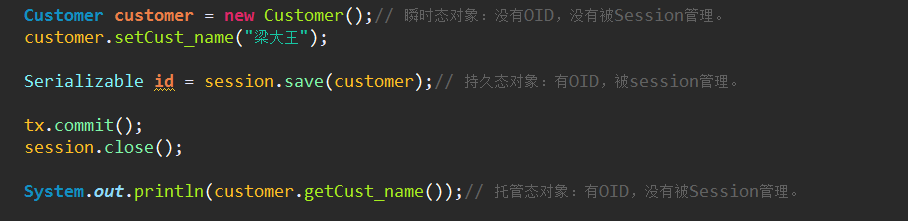
注意：在实际过程中，适用主键生成的策略一般会采用native或者identity或者sequence;

05-Hibernate持久化类的三种状态：三种状态的概述.avi

A：hibernate持久化类中状态

|  |  |  |  |
| --- | --- | --- | --- |
|  | 瞬时态(Transient) | 持久态(Persistent) | 脱管态(Detached) |
| 有没有**OID（主键）** | 没有 | 有 | 有 |
| 有没有交给**session管理（有没有调用session操作数据的方法）** | 没有 | 有 | 没有 |
| 作用 |  | 对象放入到了session的一级缓存中 |  |

例如：

  
有没有OID看是否有该Bean有没有主键，有没有交给Session管理看是否调用了Session操作数据库方法。刚new出来的对象是没有OID也没有被session管理；save()方法之后的对象是通过hibernate配置的主键生成策略来完成的OID的。通过session.close()方法关session。该对象就没有交给session管理了。

注意：持久态对象之所以能够更新数据，是因为具备了主键且交给了session的管理。

06-Hibernate持久化类的三种状态：三种状态转换.avi

A：hibernate持久化类3种状态之间的转变

|  |  |  |  |
| --- | --- | --- | --- |
| 行转列 | 瞬时态 | 持久态 | 脱管态 |
| 瞬时态 |  | save()  saveOrupdate() | setCustomerId()  设置主键 |
| 持久态 | delete() |  | close()  clear()  evict() |
| 脱管态 | setCustomerId(null)设置主键为null | saveOrUpdate()  update() |  |

这里的saveOrUpdate()可以做save()也可以做update()操作；如果该对象有id则做update()操作。如果没有id则做save()操作；

注意：持久化类状态转变只作了解，但是需要关注的是该对象有没有主键，有没有在session中操作；持久化状态对象是放在一级缓存中的。

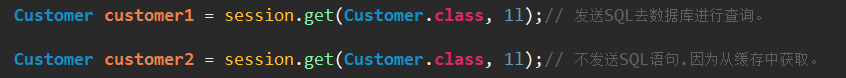
07-Hibernate的一级缓存：一级缓存的概述.avi

A: hibernate一级缓存

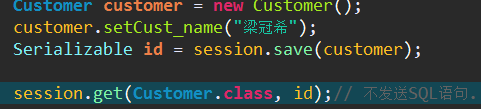
hibernate默认的是一级缓存，二级缓存需要借助第三方的缓存配置来完成；

缓存的设计的目的就是提高查询的速度。做查询操作的时候会首先去缓存中找，有没有。如果有直接从缓存中拿到，这个时候就不会去数据库查询。如果没有才会去数据库进行查询然后将查询出来的数据放入到缓存中，那么下次再查询就直接从缓存中拿了。

例如：



首先由于session缓存是不存在该id=1l的数据，所有这个时候需要去数据库查询。然后将查询该条数据放入到缓存中，所以在执行第二次get()操作的时候就直接从缓存中拿了。



当做save()操作的时候，会向session中的一级缓存中存入该条数据，再次查询的时候也会首先从缓存中取。

注意：查询数据会首先从缓存中看是否有，没有采取数据库中查询然后再存入到缓存中。插入数据会将数据放入到缓存中。

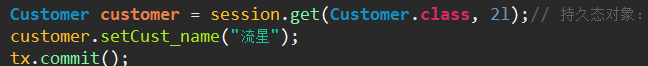
08-Hibernate的一级缓存：一级缓存的快照区.avi

A：快照设计

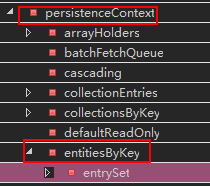
“快照”在计算机中很重要，例如，在虚拟机中virtual box中有一个设置用来给虚拟机中安装的系统设置快照，根据该快照还原到该点。快照就是将之前某一个状态保存下来，用来对比还原。快照的英文单词是：snapshot.在maven项目管理工具就有该概念。还比如window系统在升级的时候，出现错误可以还原到上一个版本上，就是根据系统的快照来还原的。

B：一级缓存中的缓存设计

1. 通过持久态bean改变缓存中的属性值(没有使用update方法)

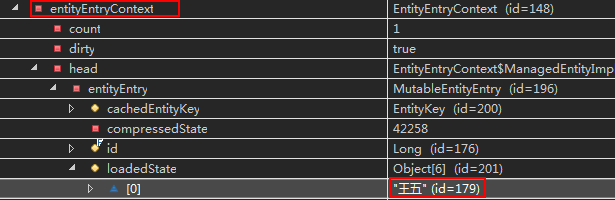


1. Session中的缓存



1. Session中的快照

在往缓存中存入的同时，快照区（前提是快照区是空的）也存了一份



当改变持久态bean的时候，只改变了缓存区，并没有再改变快照区

session.commit()操作的时候会判断缓存区中和快照区中是否一样。如果不一样就会修改表中数据；

注意：刚开始如果缓存区中没有数据，如果再做了操作之后向缓存区中存入数据，这个时候就快照一下（就是把缓存区的数据也备份一份放到快照中），一旦改变缓存区中数据，会对比缓存区和快照区看是否一样，如果不一样就会改变表中的数据；

09-Hibernate的事务管理：事务的回顾.avi

A：事务的特点

事务就是一组操作，这组操作要么全部完成，要么都不完成。遵循的是ACID特性；

原子性（Atomicity）：在一个事务中，那么所有操作都完成，要么所有操作都不完成。

一致性（Consistency）：事务执行前后，数据的完整性保持一致。

隔离性（Isolation）：如果有两个事务，运行在相同的时间内，执行相同的功能，事务的隔离性将确保每一事务在系统中认为只有该事务在使用系统。让事务串行执行，互不影响。

持久性（Durability）：在事务完成以后，该事务对数据库所作的更改便持久的保存在数据库之中，并不会被回滚

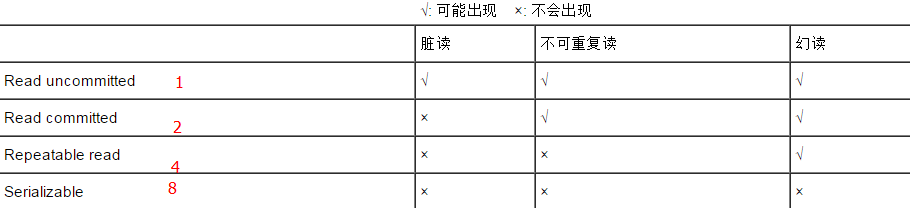
B：不考虑事务，可能存在的问题

脏读：一个事务读到另一个事务未提交的数据。

不可重复读：一个事务读到另一个事务已经提交的update数据，导致多次查询结果不一致。

虚读(幻读)：一个事务读到另一个事务已经提交的insert数据，导致多次查询结果不一致。

C：事务的隔离级别



这里的4代表设置成Repeatable read 这是mysql的默认隔离级别。oralce中默认的隔离级别是read committed；

注意：通过设置隔离级别可以防止上面的问题出现；了解事务的特性；

10-Hibernate的事务管理：与线程绑定的Session.avi

A：在业务层中绑定session

在业务层中绑定session,这样保证了一个service层中每一个dao使用的都是同一个session在hibernate.cfg.xml配置文件可以通过多种方式是想不绑定，其中比较常见的是将session绑定到本地线程上ThreadLocal中。



B：从线程中获取Session对象

之前我们获取Session对象是通过Session session=sessionFactory.openSession();

这里由于将session绑定在了本地线程上，所以可以使用另外一个方法获取

Session session=sessionFactory.getCurrentSession();



上面的获取session保证了在同一个线程中session是一致的。在service中多个dao执行所用的session是同一个session。

注意：一个session对象除了open(),还可以使用getCurrentSession()但是前提是该session需要绑定起来；

11-Hibernate的其他API：Query接口.avi

A：Query接口

1. HQL语法

HQL跟SQL有点相似，但是跟SQL语句有区别，在HQL语句中不能写表或者表中列名，需要写类或者对象属性的名字；对大小写敏感，不能使用select \* from ；

后面会其更详细介绍HQL。在Query接口创建中需要HQL.

1. Query接口获取

Query query=session.createQuery(HQL);

1. 常用的HQL语句

最简单查询语句 from Customer as customer;

select c.cust\_name from Customer as c where c.cust\_name like ?;

至于多表连接，子查询，分组查询都是可以实现的。

query.setParameter(0,”%梁%”);这里的序号是从0开始的，跟jdbc从1开始有点区别；

注意：这里只是简单地介绍了Query接口，通过Query接口我们可以完成更复杂的SQL语句去查询结果；

12-Hibernate的其他API：Criteria接口.avi

A：Criteria接口

1. Criteria接口获取

Criteria criteria=session.createCriteria(Customer.class);

Criteria是用来条件查询的

1. 使用Criteria接口完成条件查询

给Criteria接口设置过滤条件



其中Restrictions还有其他方法如eq(),in(),isEmpty()等。

给Criteria设置好了过滤条件之后，就可以获取数据了

criteria.list()获取list集合了

注意：Criteria接口跟Query接口一样，都是通过Session通过createxxx()方法来获取。Criteria接口中有一个Restrictions类用来设置过滤条件；这些接口执行完了，都可以通过该接口获取结果；

13-Hibernate的其他API：SQLQuery接口.avi

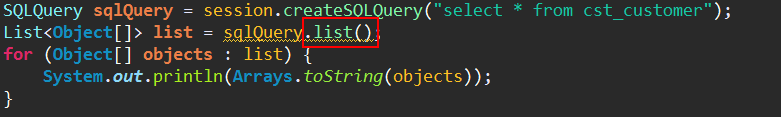
A：SQLQuery接口

1. SQLQuery接口获取

SQLQuery sqlQuery=session.createSQLQuery(sql语句);

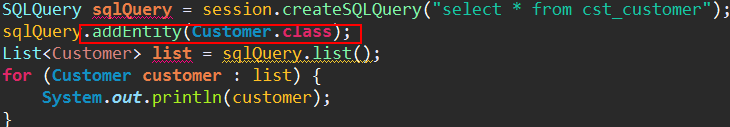
这里的sql语句就是平常我们在数据库去写的sql语句。

1. SQLQuery接口使用



这里的Object[]封装的是每一个属性的值。

如果想返回的是所有的Customer，则可以通过如下设置



通过addEntity()设置了返回的是bean对象

注意：SQLQuery接口是用SQL语句来进行查询的。在复杂的查询中，我们可以使用该接口。一般可以通过addEntity()来设置返回到是装有bean对象的list集合。

14-今天内容总结.avi

A：hibernate持久化类编写规则

要求：无参构造，属性私有化并提供set,get方法

B：hibernate主键生成策略

native,increment,sequence,

C：hibernate bean的3种状态

瞬时态 没有OID 没有交给session管理

持久态 有OID，交给了session管理

脱管态 有OID，没有交给session管理

new Bean(); 瞬时态

save() 瞬时态转持久态

update() 托管态转持久态

saveOrUpdate():没有设置主键则save(),有主键则update()

delete() 持久态转瞬时态

evict() 持久态转脱管态

clear() 持久态转脱管态

D：hibernate一级缓存和快照区

通过改变持久态bean属性，则commit提交就可以改变数据库表中数据；每次查询首先是从一级缓存中查询的，如果没有则从数据库中查询然后放入到一级缓存中；下次再查询就从缓存中拿出来。commit()会会对比缓存区中数据是否一致，如果不一致就会改变数据库表数据；

E：hibernate的其他接口api

1. Query query=session.createQuery(HQL);

query.setParameter();

1. Criteria criteria=session.createCriteria(xxx.class);

criteria.add(Restrictions.xxx())

1. SQLQuery sqlQuery=session.createSQLQuery(sql)

sqlQuery.addEntity();

每一个接口都有list()