**Hibernate框架核心接口和类**

**核心类和接口**：Hibernate的核心类和接口一共有6个，分别为:Session、SessionFactory、Transaction、Query、Criteria和Configuration。

* **Session** ：该接口负责执行被持久化对象的CRUD操作(CRUD是指在做计算处理时的增加(Create)、读取(Retrieve)（重新得到数据）、更新(Update)和删除(Delete)几个单词的首字母简写)
* **SessionFactory** ：负责初始化Hibernate,充当数据存储源的代理，负责创建Session对象
* **Transaction** ： 该接口是对实际事务实现的一个抽象，隔离了底层不同事务实现
* **Query** ：Query接口让你方便地对数据库及持久对象进行查询，有两种表达方式：HQL语言或本地数据库的SQL语句
* **Criteria** ：Criteria接口与Query接口非常类似，允许创建并执行面向对象的标准化查询
* **Configuration** ：Configuration 类的作用是对Hibernate 进行配置，以及对它进行启动，Configuration 类的实例首先定位映射文档的位置，读取这些配置，然后创建一个SessionFactory对象

Configuration 类负责管理Hibernate 的配置信息。它包括如下内容：

* Hibernate运行的底层信息：数据库的URL、用户名、密码、JDBC驱动类，数据库Dialect,数据库连接池等。

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<!-- 指定连接数据库所用的驱动 -->

<property name="connection.driver\_class">com.mysql.jdbc.Driver</property>

<!-- 指定连接数据库的url，其中hibernate是本应用连接的数据库名 -->

<property name="connection.url">jdbc:mysql://localhost/hibernate\_test</property>

<!-- 指定连接数据库的用户名 -->

<property name="connection.username">root</property>

<!-- 指定连接数据库的密码 -->

<property name="connection.password">cheng</property>

<!-- 指定连接池里最大连接数 -->

<property name="hibernate.c3p0.max\_size">20</property>

<!-- 指定连接池里最小连接数 -->

<property name="hibernate.c3p0.min\_size">1</property>

<!-- 指定连接池里连接的超时时长 -->

<property name="hibernate.c3p0.timeout">5000</property>

<!-- 指定连接池里最大缓存多少个Statement对象 -->

<property name="hibernate.c3p0.max\_statements">100</property>

<property name="hibernate.c3p0.idle\_test\_period">3000</property>

<property name="hibernate.c3p0.acquire\_increment">2</property>

<property name="hibernate.c3p0.validate">true</property>

<!-- 指定数据库方言 -->

<property name="dialect">org.hibernate.dialect.MySQL5InnoDBDialect</property>

<!-- 根据需要自动创建数据表 -->

<property name="hbm2ddl.auto">update</property><!--①-->

<!-- 显示Hibernate持久化操作所生成的SQL -->

<property name="show\_sql">true</property>

<!-- 将SQL脚本进行格式化后再输出 -->

<property name="hibernate.format\_sql">true</property>

<!-- 避免这个错误信息Disabling contextual LOB creation as createClob() method threw error :java.lang.reflect.InvocationTargetException -->

<property name="hibernate.temp.use\_jdbc\_metadata\_defaults">false</property>

<!-- 罗列所有持久化类的类名 -->

<mapping class="com.wechat.entity.po.User"/>

<mapping class="com.wechat.entity.po.Person"/>

</session-factory>

</hibernate-configuration>

**SessionFactory**: 应用程序从SessionFactory（会话工厂）里获得Session(会话)实例。它在多个应用线程间进行共享。通常情况下，整个应用只有唯一的一个会话工厂——例如在应用初始化时被创建。然而，如果你使用Hibernate访问多个数据库，你需要对每一个数据库使用一个会话工厂。

SessionFactory在Hibernate中实际起到了一个缓冲区的作用，它缓冲了Hibernate自动生成的SQL语句和一些其它的映射数据，还缓冲了一些将来有可能重复利用的数据。

**Session**:

* Session接口是Hibernate对外提供的操作数据库的关键接口，它承担了管理持久对象状态的职能，通过提供基本的创建，读取，更新和删除等接口方法来实现对对象的持久化操作．
* Session不是线程安全的，它代表与数据库之间的一次操作，它的概念介于Connection和Transaction之间。
* Session也称为持久化管理器，因为它是与持久化有关的操作接口。
* Session通过SessionFactory打开，在所有的工作完成后，需要关闭。
* 它与Web层的HttpSession没有任何关系。

Query（查询）接口允许你在数据库上执行查询并控制查询如何执行。查询语句使用HQL或者本地数据库的SQL方言编写。

* **瞬时对象** 瞬时对象是被Hibernate持久化之前，由java的new 操作创建的对象．此时候该对象没有与Hibernate的Session发生关联．瞬时对象没有被持久化到数据库中，也没有被Hibernate设置标识．
* **持久对象** 持久对象定义为关联到数据库记录的对象实例．同时具有持久化标识，该标识将持久对象和数据库记录的主键关联．
* **脱管对象** 脱管对象定义为持久对象的延续，当持久对象所关联的Hibernate Session被关闭后，即调用Session的close()方法，该持久对象就转变为脱管对象.此时的对象状态不再被Hibernate的Session所管理，对象属性和数据库记录不再保持同步．

**Session操作对象的方法**:

Session对象是Hibernate对外接口中最常用的对象，它提供以下服务

１．基本的增删改查操作

２．查询操作

３．事务控制

４．事务级别的缓存管理(一级缓存)

Session通过SessionFactory创建出来后进行对象的持久化操作．Session接口主要提供了以下几种方法操作对象：

1. save()

2. get()

3. update()

4. delete()

5.saveOrUpdate()

* Session的load()和get()方法都能根据给定的OID从数据库中加载一个持久化对象，这两个方法的区别在于：当数据库中不存在与OID对应的记录时，load()方法抛出net.sf.ObjectNotFoundException异常，而get()方法返回null.
* 由get()、load()或其他查询方法返回的对象都位于当前Session的缓存中，因此修改了持久化对象的属性后，当Session清理缓存时，会根据持久化对象的属性变化来同步更新数据库
* 脱离了Session管理的对象称为脱管对象，此时对该对象的属性修改不能够反映到数据库中，这时候需要调用update方法将该对象重新关联到Session.update()方法能够通知Session生成一条Update语句同步数据库．

customer.setName(“customer1”);

Transaction tx=session.beginTransaction();

session.update(customer);

//提交

tx.commit();

session.close();

delete()方法用于从数据库中删除一个实体所对应的数据，如果传入的对象是持久化对象，那么当清理缓存时，就会执行delete操作。如果传入的是脱管对象，那么首先会使该对象与session相关联，然后当清理缓存时，再执行delete操作

**Hibernate框架常用注解**：

* **@Entity** ：将一个类声明为一个实体bean(即一个持久化POJO类)
* **@Id** ：注解声明了该实体bean的标识属性（对应表中的主键）
* **@Table** ：注解声明了该实体bean映射指定的表（table）,目录（catalog）和schema的名字
* **@GeneratedValue** ：注解声明了主键的生成策略。
* **@OneToOne** ：设置一对一个关联
* **@ManyToOne** ：设置多对一关联,该注解标注的属性通常是数据库表的外键
* **@OneToMany** ：描述一个一对多的关联,该属性应该为集合类型,在数据库中并没有实际字段
* **@JoinColumn**：描述的是一个关联字段，用于被@ManyToOne注解的字段，并且为外键

**@Column 注解的相关属性**：

* name  可选，列名（默认值是属性名）
* unique 可选，是否在该列上设置唯一约束（默认值false）
* nullable 可选，是否设置该列的值可以为空（默认值false）
* length 可选，列长度（默认值255）
* precision 可选，列十进制精度（decimal precision)(默认值0）
* scale 可选，如果列十进制数值范围（decimal scale）可用，在此设置（默认值0）
* precision属性和scale属性表示精度时，当字段类型为double时，

precision表示数值的总长度，scale表示小数点所占的位数。

* insertable 可选，该列是否作为生成的insert语句中的一个列（默认值true）
* updatable 可选，该列是否作为生成的update语句中的一个列（默认值true）

**hibernate.dialect ;一个Hibernate Dialect类名允许Hibernate针对特定的关系数据库生成优化的SQL. 取值 full.classname.of.Dialect**

**hibernate.show\_sql ;输出所有SQL语句到控制台. 有一个另外的选择是把org.hibernate.SQL这个log category设为debug。 eg. true | false**

**hibernate.format\_sql 在log和console中打印出更漂亮的SQL。 取值 true | false**

**hibernate.default\_schema 在生成的SQL中, 将给定的schema/tablespace附加于非全限定名的表名上. 取值 SCHEMA\_NAME**

**hibernate.default\_catalog 在生成的SQL中, 将给定的catalog附加于非全限定名的表名上. 取值 CATALOG\_NAME**

**hibernate.session\_factory\_name SessionFactory创建后，将自动使用这个名字绑定到JNDI中. 取值 jndi/composite/name**

**hibernate.max\_fetch\_depth 为单向关联(一对一, 多对一)的外连接抓取（outer join fetch）树设置最大深度. 值为0意味着将关闭默认的外连接抓取. 取值 建议在0到3之间取值**

**hibernate.default\_batch\_fetch\_size 为Hibernate关联的批量抓取设置默认数量. 取值 建议的取值为4, 8, 和16**

**hibernate.default\_entity\_mode 为由这个SessionFactory打开的所有Session指定默认的实体表现模式. 取值 dynamic-map, dom4j, pojo**

**hibernate.order\_updates 强制Hibernate按照被更新数据的主键，为SQL更新排序。这么做将减少在高并发系统中事务的死锁。 取值 true | false**

**hibernate.generate\_statistics 如果开启, Hibernate将收集有助于性能调节的统计数据. 取值 true | false**

**hibernate.use\_identifer\_rollback 如果开启, 在对象被删除时生成的标识属性将被重设为默认值. 取值 true | false**

**hibernate.use\_sql\_comments 如果开启, Hibernate将在SQL中生成有助于调试的注释信息, 默认值为false. 取值 true | false**

工作中遇到的：

1. 实体类中使用了@Table注解后，想要添加表中不存在字段，就要使用@Transient这个注解了。
2. 作用是json序列化时将java bean中的一些属性忽略掉,序列化和反序列化都受影响。一般在密码属性上和使用@Transient的属性；