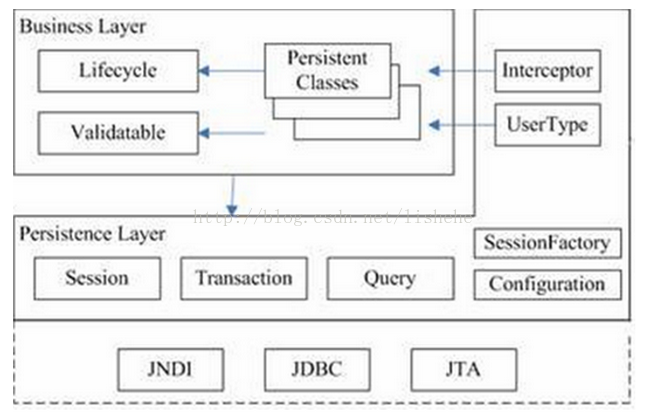
hibernate五大核心接口

Hibernate有五大核心接口，分别是：Session、SessionFactory、Transaction、Query、Configuration。这五个接口构成了Hibernate运行的基本要素，可以执行存取，持久化，事务管理等操作。这五个接口可以位于系统的业务逻辑层和持久层。下面是一张Hibernate的关系图：



## Session接口

Session接口对于Hibernate开发人员来说是一个最重要的接口。然而在Hibernate中，实例化的Session是一个轻量级的类，创建和销毁它都不会占用很多资源。这在实际项目中确实很重要，因为在客户程序中，可能会不断地创建以及销毁Session对象，如果Session的开销太大，会给系统带来不良影响。但是Session对象是非线程安全的，因此在你的设计中，最好是一个线程只创建一个Session对象。Session可以看作介于数据连接与事务管理的一种中间接口。我们可以将Session想象成一个持久对象的缓冲区，Hibernate能检测到这些持久对象的改变，并及时刷新数据库。我们有时也称Session是一个持久层管理器，因为它包含这一些持久层相关的操作，诸如存储持久对象至数据库，以及从数据库获得它们。需要注意的是，Hibernate的session不同于JSP应用中的HttpSession。当我们使用session这个术语时，我们指的是Hibernate中session，而我们以后会将HttpSession对象称为用户session。

## SessionFactory接口

SessionFactory接口负责初始化Hibernate。它充当数据存储源的代理，并负责创建Session对象。这里用到了工厂模式。需要注意的是SessionFactory并不是轻量级的，因为一般情况下，一个项目通常只需要一个SessionFactory就够，当需要操作多个数据库时，可以为每个数据库指定一个SessionFactory。

## Transaction接口

Transaction接口负责事务相关的操作，一般在Hibernate的增删改中出现，但是使用Hibernate的人一般使用Spring去管理事务。

## Query接口

Query负责执行各种数据库查询。它可以使用HQL语言或者SQL语言两种表达式。它的返回值一般是List。需要自己转换。

## Configuration接口

Configuration对象用于配置并根据配置启动Hibernate。Hibernate应用通过Configuration实例来指定对象——关系映射文件的位置或者动态配置Hibernate的属性，然后创建SessionFactory实例。

Spring对Hibernate的事务管理

## 声明式事务配置

1、声明式事务配置

（1）配置事务管理器；

（2）事务的传播特性；

（3）哪些类哪些方法使用事务。

<!-- 配置事务管理器 指定其作用的sessionFactory把事务交给Spring去处理 -->

<bean id="transactionManager" class="org.springframework.orm.hibernate3.HibernateTransactionManager">

<property name="sessionFactory">

<ref bean="sessionFactory"/>

</property>

</bean>

<!-- 配置事务的传播特性 -->

<tx:advice id="txAdvice" transaction-manager="transactionManager">

<tx:attributes>

<tx:method name="save\*" propagation="REQUIRED"/>

<tx:method name="delete\*" propagation="REQUIRED"/>

<tx:method name="update\*" propagation="REQUIRED"/>

<tx:method name="get\*" read-only="true" propagation="NOT\_SUPPORTED"/>

<tx:method name="\*" read-only="true"/>

</tx:attributes>

</tx:advice>

<!-- 那些类的哪些方法参与事务 -->

<aop:config>

<aop:pointcut id="allServiceMethod" expression="execution(\* com.coe.service.\*.\*(..))"/>

<aop:advisor pointcut-ref="allServiceMethod" advice-ref="txAdvice"/>

</aop:config>

我们在配置事务的时候，我们一般是把事务边界设置到service层，也就是你的业务逻辑层，因为我们很多时候都是在我们的业务逻辑层来完成我们一 些列的数据操作，如果放到Dao数据层，其粒度太小了。另外，如果我们把事务配置在业务逻辑层的话，对我们的二级缓存也是有好处的，这个大家以后实际操作 的时候会发现。

2、编写业务逻辑方法

这时候我们就可以在我们业务逻辑层用HibernateTemplate里面提供的数据操作方法来编写我们的业务逻辑方法了，当然我们的方法必须要 是以我们事务配置里面配置的一样，用save，delete，update，get做我们的方法的开头。需要注意的是，默认情况下运行期异常才会回滚（包 括继承了RuntimeException子类），普通异常是不会滚的。

最后我们顺便总结一下事务的几种传播特性：

1. PROPAGATION\_REQUIRED: 如果存在一个事务，则支持当前事务。如果没有事务则开启；

2. PROPAGATION\_SUPPORTS: 如果存在一个事务，支持当前事务。如果没有事务，则非事务的执行；

3. PROPAGATION\_MANDATORY: 如果已经存在一个事务，支持当前事务。如果没有一个活动的事务，则抛出异常；

4. PROPAGATION\_REQUIRES\_NEW: 总是开启一个新的事务。如果一个事务已经存在，则将这个存在的事务挂起；

5. PROPAGATION\_NOT\_SUPPORTED: 总是非事务地执行，并挂起任何存在的事务；

6. PROPAGATION\_NEVER: 总是非事务地执行，如果存在一个活动事务，则抛出异常；

7. PROPAGATION\_NESTED：如果一个活动的事务存在，则运行在一个嵌套的事务中. 如果没有活动事务, 则按TransactionDefinition.PROPAGATION\_REQUIRED 属性执行。