为了保证数据的一致性，在编程的时候往往需要引入事务这个概念。事务有4个特性：原子性、一致性、隔离性、持久性。

         事务的种类有两种：编程式事务和声明式事务。编程式事务就是将事务处理放在程序中，而声明式事务则是通过配置文件或者注解进行操作。

         在[**spring**](http://lib.csdn.net/base/javaee)中有声明式事务的概念，通过和[**hibernate**](http://lib.csdn.net/base/javaee)类似框架的集成，可以很好的完成声明式事务。

         其实，不论在Spring中有几种配置Hibernate事务的方法，都逃不出一下几条：

         1.配置SessionFactory

         2.配置事务容器

         3.配置事务规则

         4.配置事务入口

         后面一共为大家提供4种配置Hibernate事务的方法。

         首先说下配置SessionFactory，配置SessionFactory有两种方式，一种是通过配置hibernate.cfg.xml文件的位置来配置SessionFactory，另一种就是在Spring配置文件中，手动配置数据源。

         下面是两种配置SessionFactory的方式（第二种配置需要额外引入两个包：commons-dbcp、commons-pool）

<!-- 1、第一种配置SessionFactory的方式 -->

<beanid="sessionFactory"

    class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">

    <propertyname="configLocation"value="classpath:hibernate.cfg.xml"/>

</bean>

<!-- 2、第二种配置SessionFactory的方式 -->

<!-- 2.1配置数据源 -->

<beanid="dataSource"class="org.apache.commons.dbcp.BasicDataSource"

    destroy-method="close">

 <propertyname="driverClassName"value="com.mysql.jdbc.Driver"></property>

 <propertyname="url"value="jdbc:mysql://localhost:3306/hibernate\_cache"></property>

    <propertyname="username"value="root"></property>

    <propertyname="password"value="admin"></property>

</bean>

<!-- 2.2、配置SessionFactory -->

<beanid="sessionFactory"

    class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">

    <propertyname="dataSource"ref="dataSource"></property>

    <propertyname="hibernateProperties">

        <props>

            <propkey="hibernate.hbm2ddl.auto">update</prop>

        </props>

    </property>

    <propertyname="mappingLocations">

        <list>

            <value>classpath:实体对应xml的路径</value>

        </list>

    </property>

</bean>

         至此Hibernate就成功的将SessionFactory交给了Spring来管理。现在再来看Spring是怎样管理Hibernate事务的吧。

         第一种方式，利用tx标签配置事务。

<!-- 配置事务容器 -->

<beanid="transactionManager"

    class="org.springframework.orm.hibernate3.HibernateTransactionManager">

    <propertyname="sessionFactory"ref="sessionFactory"/>

</bean>

<!-- 定义事务规则 -->

<tx:adviceid="txAdvice"transaction-manager="transactionManager">

    <tx:attributes>

        <tx:methodname="add\*"propagation="REQUIRED"/>

        <tx:methodname="modify\*"propagation="REQUIRED"/>

        <tx:methodname="del\*"propagation="REQUIRED"/>

        <tx:methodname="\*"propagation="REQUIRED"read-only="true"/>

    </tx:attributes>

</tx:advice>

<!-- 定义事务入口 -->

<aop:config>

   <aop:pointcutid="allDaoMethod"expression="execution(\* com.jianxin.dao.\*.\*(..))"  />

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

</aop:config>

         第二种，用代理进行配置

<!-- 配置事务容器 -->

<beanid="transactionManager"

    class="org.springframework.orm.hibernate3.HibernateTransactionManager">

    <propertyname="sessionFactory"ref="sessionFactory"/>

</bean>

<!-- 定义事务规则 -->

<beanid="transactionProxy"

    class="org.springframework.transaction.interceptor.TransactionProxyFactoryBean"

    abstract="true">

    <propertyname="transactionManager"ref="transactionManager"/>

    <propertyname="transactionAttributes">

        <props>

            <!-- ，回滚为-，不回滚为+ -->

            <propkey="add\*">PROPAGATION\_REQUIRED,-Exception</prop>

            <propkey="modify\*">PROPAGATION\_REQUIRED,+MyException</prop>

            <propkey="del\*">PROPAGATION\_REQUIRED</prop>

            <propkey="\*">READONLY</prop>

        </props>

    </property>

</bean>

<!-- 定义事务入口 -->

<beanid="userDaoProxy"parent="transactionProxy">

    <propertyname="target"ref="userDao"></property>

</bean>

        第三种，利用拦截器

<!-- 配置事务容器 -->

<beanid="transactionManager"

    class="org.springframework.orm.hibernate3.HibernateTransactionManager">

    <propertyname="sessionFactory"ref="sessionFactory"/>

</bean>

<!-- 定义事务规则 -->

<beanid="transactionInterceptor"

    class="org.springframework.transaction.interceptor.TransactionInterceptor">

    <propertyname="transactionManager"ref="transactionManager"/>

    <propertyname="transactionAttributes">

        <props>

            <!-- 回滚为-，不回滚为+ -->

            <propkey="add\*">PROPAGATION\_REQUIRED,-Exception</prop>

            <propkey="modify\*">PROPAGATION\_REQUIRED,+MyException</prop>

            <propkey="del\*">PROPAGATION\_REQUIRED</prop>

            <propkey="\*">READONLY</prop>

        </props>

    </property>

</bean>

<!-- 定义事务入口 -->

<beanid="proxyFactory"

    class="org.springframework.aop.framework.autoproxy.BeanNameAutoProxyCreator">

    <propertyname="interceptorNames">

        <list>

            <value>transactionInterceptor</value>

        </list>

    </property>

    <propertyname="beanNames">

        <list>

            <value>\*Dao</value>

        </list>

    </property>

    <!--使用接口时-->

    <property name="proxyTargetClass">  
<value>true</value>  
    </property>

</bean>

         第四种，利用注解。

<!-- 开户事务注解功能 -->

<tx:annotation-driven transaction-manager="transactionManager"/>

         首先，在配置文件中写入下面语句，打开注解功能

         然后用@Transactional对类或者方法进行标记，如果标记到类上，那么次类中所有方法都进行事务回滚处理，在类中定义Transactional的时候，它有propagation、rollbackFor、noRollbackFor等属性，此属性是用来定义事务规则，而定义到哪这个就是事务入口。

         纵观以上四种在Spring中配置Hibernate事务的方法，其核心都是一样的，不同的只是实现的方式而已。所以看到这，这篇博文中你只需要记住4句话，就可以轻松理解在Spring中配置Hibernate事务的核心：

         1.配置SessionFactory

         2.配置事务容器

         3.配置事务规则

         4.配置事务入口