Spring是SSH中的管理员，负责管理其它框架，协调各个部分的工作。今天一起学习一下Spring的事务管理。Spring配置文件中关于事务配置总是由三个组成部分，分别是DataSource、TransactionManager和代理机制这三部分，无论哪种配置方式，一般变化的只是代理机制这部分。 DataSource、TransactionManager这两部分只是会根据数据访问方式有所变化，比如使用Hibernate进行数据访问时，DataSource实际为**SessionFactory**，TransactionManager的实现为**HibernateTransactionManager**。
下面一起看看三种声明式事务的具体配置：

公共配置

```
<!-- 配置sessionFactory -->
<bean id="sessionFactory"
class="org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBean">
    <property name="configLocation">
        <value>classpath:config/hibernate.cfg.xml</value>
    </property>
    <property name="packagesToScan">
        <list>
            <value>com.entity</value>
        </list>
    </property>
</bean>

<!-- 配置事务管理器（声明式的事务） -->
<bean id="transactionManager"
class="org.springframework.orm.hibernate3.HibernateTransactionManager">
    <property name="sessionFactory" ref="sessionFactory"></property>
</bean>

<!-- 配置DAO -->
<bean id="userDao" class="com.dao.UserDaoImpl">
    <property name="sessionFactory" ref="sessionFactory"></property>
</bean>
```

第一种，使用tx标签方式

```
<!-- 第一种配置事务的方式 ，tx-->
<tx:advice id="txadvice" transaction-manager="transactionManager">
    <tx:attributes>
        <tx:method name="add*" propagation="REQUIRED" rollback-for="Exception" />
        <tx:method name="modify*" propagation="REQUIRED" rollback-for="Exception"
/>
        <tx:method name="del*" propagation="REQUIRED" rollback-for="Exception"/>
```

```
        <tx:method name="*" propagation="REQUIRED" read-only="true"/>
    </tx:attributes>
</tx:advice>

<aop:config>
    <aop:pointcut id="daoMethod" expression="execution(* com.dao.*.*(..))"/>
    <aop:advisor pointcut-ref="daoMethod" advice-ref="txadvice"/>
</aop:config>
```

expression="execution(*com.dao..(..))*"
*其中第一个*代表返回值，第二*代表dao下子包，第三个*代表方法名，“（..）”代表方法参数。

Spring事务类型详解：

- PROPAGATION_REQUIRED--支持当前事务，如果当前没有事务，就新建一个事务。这是最常见的选择。
- PROPAGATION_SUPPORTS--支持当前事务，如果当前没有事务，就以非事务方式执行。
- PROPAGATION_MANDATORY--支持当前事务，如果当前没有事务，就抛出异常。
- PROPAGATION_REQUIRES_NEW--新建事务，如果当前存在事务，把当前事务挂起。
- PROPAGATION_NOT_SUPPORTED--以非事务方式执行操作，如果当前存在事务，就把当前事务挂起。
- PROPAGATION_NEVER--以非事务方式执行，如果当前存在事务，则抛出异常。
- PROPAGATION_NESTED--如果当前存在事务，则在嵌套事务内执行。如果当前没有事务，则进行与PROPAGATION_REQUIRED类似的操作。