链滴

事务管理---声明式事务管理spring配置方式（二）

作者：oldcaptain

原文链接：https://ld246.com/article/1388649010302

来源网站：链滴

<h2>引言：</h2>
      接上篇文章的探讨。初步将事务配置分为编程式事务管理，和声明式务管理，两种事务管理的区别，已经在上篇文章中做了简要说明。下来将详细说明下spring的事务配方式。看了很多资料，也结合我们实际项目中的配置方式，大致分为四五种。
<h2>使用拦截器方式：</h2>
<pre class="brush: xml"><?xml version="1.0" encoding="UTF-8"?gt;
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
 "http://www.springframework.org/dtd/spring-beans.dtd">
<beans>

<bean id="sqlMapClient" class="org.springframework.orm.ibatis.SqlMaplientFactoryBean">
    <property name="configLocation">
        <value>classpath:sqlmap-config.xml</value>
    </property>
    <property name="dataSource" ref="dataSource" />
</bean>

<bean id="transactionInterceptor"
    class="org.springframework.transaction.interceptor.TransactionInterceptor"&gt

    <property name="transactionManager">
        <ref bean="transactionManager" />
    </property>
    <property name="transactionAttributes">
     <props>
      <prop key="remove*">PROPAGATION_REQUIRED</prop>
      <prop key="add*">PROPAGATION_REQUIRED</prop>
      <prop key="modify*">PROPAGATION_REQUIRED</prop>
      <prop key="find*">PROPAGATION_REQUIRED,readOnly</prop>
     </props>
    </property>
</bean>
<!-- 自动代理 -->
<bean id="autoproxy"
    class="org.springframework.aop.framework.autoproxy.BeanNameAutoProxyCreatorquot;>
    <property name="beanNames">
        <list>
            <value>*Service</value>
        </list>
    </property>
    <property name="interceptorNames">
        <list>
            <value>transactionInterceptor</value>
        </list>
    </property>
</bean>

</beans>
</pre>

<h2>使用tx标签配置的拦截器：</h2>
<pre class="brush: xml"><?xml version="1.0" encoding="utf-8"?&
t;
<!--基本配置信息 -->
<beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi=&
uot;http://www.w3.org/2001/XMLSchema-instance" xmlns:tx="http://www.spring
ramework.org/schema/tx"
    xmlns:aop="http://www.springframework.org/schema/aop" xmlns:jee="
ttp://www.springframework.org/schema/jee"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/s
ring-tx-2.0.xsd
    http://www.springframework.org/schema/aop http://www.springframework.org/schema/a
p/spring-aop-2.0.xsd
    http://www.springframework.org/schema/jee http://www.springframework.org/schema/jee
spring-jee-2.0.xsd"
    default-lazy-init="true" default-merge="false">
    <!--事务管理 -->
    <bean id="transactionManager" class="org.springframework.orm.hibe
nate3.HibernateTransactionManager">
        <property name="sessionFactory" ref="sessionFactory" />
    </bean>
    <tx:annotation-driven transaction-manager="transactionManager"
        proxy-target-class="true" />
    <!-- spring声明式事务的配置，以下为spring的AOP事务管理的增强部分 -->
    <tx:advice id="tx-Advice" transaction-manager="transactionManager
quot;>
        <tx:attributes>
            <!-- 需要由交给spring的aop来进行代理的方法的集合，如果应用有自己的方法需有由spri
g来进行事务控制必须添加方法 -->
            <!-- 读取数据方法，一般采用只读事务 -->
            <tx:method name="get*" isolation="DEFAULT" propagatio
="SUPPORTS" read-only="true" />
            <tx:method name="load*" isolation="DEFAULT" propagati
n="SUPPORTS" read-only="true" />
            <tx:method name="select*" isolation="DEFAULT" propagat
on="SUPPORTS" read-only="true" />
            <tx:method name="query*" isolation="DEFAULT" propagat
on="SUPPORTS" read-only="true" />
            <tx:method name="find*" isolation="DEFAULT" propagati
n="SUPPORTS" read-only="true" />
            <tx:method name="list*" isolation="DEFAULT" propagatio
="SUPPORTS" read-only="true" />
            <tx:method name="criteria*" isolation="DEFAULT" propaga
ion="SUPPORTS" read-only="true" />
            <tx:method name="*" isolation="DEFAULT" propagation=
quot;SUPPORTS" read-only="true" />
            <!--其他方法，如save，update，insert等对数据库进行写入操作的方法，当产生Exceptio
进行回滚 -->
            <tx:method name="init*" isolation="DEFAULT" read-only=
quot;false" propagation="REQUIRED" rollback-for="Exception"
/>
            <tx:method name="insert*" isolation="DEFAULT" read-onl

```
=&quot;false&quot; propagation=&quot;REQUIRED&quot; rollback-for=&quot;Exception&quot; /&gt;
            &lt;tx:method name=&quot;update*&quot; isolation=&quot;DEFAULT&quot; read-only=&quot;false&quot; propagation=&quot;REQUIRED&quot; rollback-for=&quot;Exception&quot; /&gt;
            &lt;!-- 调用接口记录日志的事物 --&gt;
            &lt;tx:method name=&quot;createPNR&quot; isolation=&quot;DEFAULT&quot; read-only=&quot;false&quot; propagation=&quot;REQUIRED&quot; rollback-for=&quot;Exception&quot; /&gt;
            &lt;tx:method name=&quot;queryQueryFlightInfos&quot; isolation=&quot;DEFAULT&quot; read-only=&quot;false&quot; propagation=&quot;REQUIRED&quot; rollback-for=&quot;Exception&quot; /&gt;
            &lt;tx:method name=&quot;cancelPNR&quot; isolation=&quot;DEFAULT&quot; read-only=&quot;false&quot; propagation=&quot;REQUIRED&quot; rollback-for=&quot;Exception&quot; /&gt;
            &lt;!-- 对外提供接口记录日志的事物 --&gt;
            &lt;tx:method name=&quot;getPNR&quot; isolation=&quot;DEFAULT&quot; read-only=&quot;false&quot; propagation=&quot;REQUIRED&quot; rollback-for=&quot;Exception&quot; /&gt;
            &lt;tx:method name=&quot;doThirdPata&quot; isolation=&quot;DEFAULT&quot; rea-only=&quot;false&quot; propagation=&quot;REQUIRED&quot; rollback-for=&quot;Exception&quot; /&gt;
        &lt;/tx:attributes&gt;
    &lt;/tx:advice&gt;
    &lt;aop:config&gt;
        &lt;aop:advisor advice-ref=&quot;tx-Advice&quot;
            pointcut=&quot;execution(* com.newtouch.platform..service..*Service*.*(..))
            || execution(* com.test.service..*Service*.*(..))
            || execution(* cn.com.besttone.reservation.service..*Service*.*(..))&quot;/&gt;
    &lt;/aop:config&gt;
&lt;/beans&gt;
</pre>
<h2>使用注解方式</h2>
<pre class="brush: java">    &lt;context:annotation-config /&gt;
    &lt;context:component-scan base-package=&quot;com.bluesky&quot; /&gt;


&lt;tx:annotation-driven transaction-manager=&quot;transactionManager&quot; /&gt;

&lt;bean id=&quot;sessionFactory&quot; class=&quot;org.springframework.orm.hibernate3.ocalSessionFactoryBean&quot;&gt;
    &lt;property name=&quot;configLocation&quot; value=&quot;classpath:hibernate.cfg.xml quot; /&gt;
    &lt;property name=&quot;configurationClass&quot; value=&quot;org.hibernate.cfg.AnnottionConfiguration&quot; /&gt;
&lt;/bean&gt;

&lt;!-- 定义事务管理器（声明式的事务） --&gt;
&lt;bean id=&quot;transactionManager&quot; class=&quot;org.springframework.orm.hibernte3.HibernateTransactionManager&quot;&gt;
    &lt;property name=&quot;sessionFactory&quot; ref=&quot;sessionFactory&quot; /&gt;
&lt;/bean&gt;
```

此时在DAO上需加上@Transactional注解，如下：

@Transactional

@Component("userDao")

public class UserDaoImpl extends HibernateDaoSupport implements UserDao {

public List<User> listUsers() {

return this.getSession().createQuery("from User
quot;).list();

}

.....

}</pre>

<h2>共享代理类</h2>
<pre class="brush: xml"><bean id="sessionFactory" class="org.springfr
mework.orm.hibernate3.LocalSessionFactoryBean">
    <property name="configLocation" value="classpath:hibernate.cfg.
ml" />
    <property name="configurationClass" value="org.hibernate.cfg.An
otationConfiguration" />
  </bean>

<!-- 定义事务管理器（声明式的事务） -->
<bean id="transactionManager" class="org.springframework.orm.hibern
te3.HibernateTransactionManager">
  <property name="sessionFactory" ref="sessionFactory" />
</bean>

<bean id="transactionBase" class="org.springframework.transaction.inte
ceptor.TransactionProxyFactoryBean" lazy-init="true" abstract="true
quot;>
  <!-- 配置事务管理器 -->
  <property name="transactionManager" ref="transactionManager&qu
t; />
  <!-- 配置事务属性 -->
  <property name="transactionAttributes">
    <props>
      <prop key="*">PROPAGATION_REQUIRED</prop>
    </props>
  </property>
</bean>

<!-- 配置DAO -->
<bean id="userDaoTarget" class="com.bluesky.spring.dao.UserDaoImpl
quot;>
  <property name="sessionFactory" ref="sessionFactory" />
</bean>

<bean id="userDao" parent="transactionBase">
  <property name="target" ref="userDaoTarget" />
</bean></pre>

<p>以上几种方式在项目中比较常用，了解这些配置，以及按照规范对应项目中的编码就没什么大的

题。</p>