@RunWith(SpringJUnit4ClassRunner.**class**)

@ContextConfiguration("classpath:sp/test/jdbc/applicationContext.xml")

**public** **class** TestJDBC {

@Transactional(propagation = Propagation.REQUIRED, readOnly = **true**)

@Test

**public** **void** test() **throws** Exception {

}

@Test

**public** **void** test2() {

// 编程式事务管理

DefaultTransactionDefinition td = **new** DefaultTransactionDefinition();

td.setIsolationLevel( TransactionDefinition.ISOLATION\_READ\_COMMITTED );

td.setPropagationBehavior( TransactionDefinition.PROPAGATION\_REQUIRED );

td.setTimeout( 20 );

TransactionStatus t = transactionManager.getTransaction( td );

**try** {

jt.execute( "insert into b1 values(6,'123',NULL)" );

Object s1 = t.createSavepoint();

**try** {

jt.execute( "..." );

} **catch** (Exception e) {

t.rollbackToSavepoint( s1 );

}

} **catch** (Exception e) {

t.setRollbackOnly();

**throw** **new** RuntimeException( e );

} **finally** {

transactionManager.commit( t );

}

}

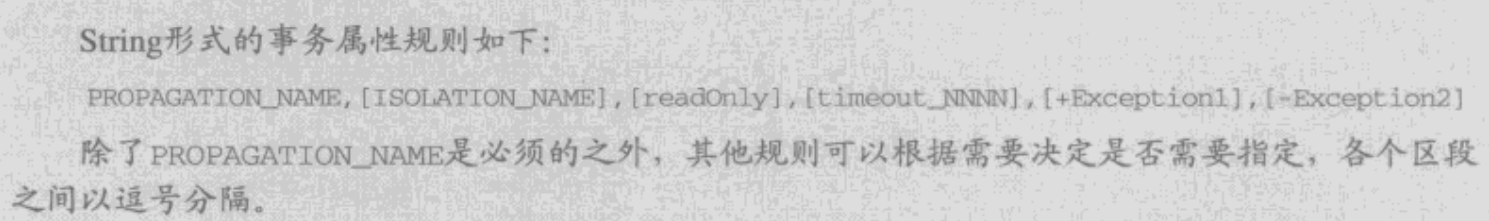
@Resource

**private** PlatformTransactionManager transactionManager;

@Resource

**private** JdbcTemplate jt;

}



配置xml来事务管理 这里用的是hibernate

首先配置文件里要有 dataSource sessionFactory

方法0 应该是最基本的方法 其他方法好像都最终是通过它的?

数据源

事务拦截器

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

<property name="transactionManager" ref="transactionManager" />

<property name="transactionAttributes">

<props>

<prop key="\*">PROPAGATION\_REQUIRED</prop>

</props>

</property>

</bean>

//目标对象

<bean id="singerTarget" class="sp.tran0.Singer">

<property name="sessionFactory" ref="sessionFactory" />

</bean>

代理对象

<bean id="singer" class="org.springframework.aop.framework.ProxyFactoryBean">

<property name="target" ref="singerTarget" />

<property name="proxyInterfaces">

<array>

<value>sp.tran0.Performer</value>

</array>

</property>

<property name="interceptorNames">

<array>

<value>transactionInterceptor</value>

</array>

</property>

</bean>

作为方法0的一点扩展可以使用BeanNameAutoProxyCreator

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

<property name="interceptorNames">

<value>transactionInterceptor</value>

</property>

<property name="beanNames">

<value>singer</value> 原来的bean名字从singerTarget改回singer

</property>

</bean>

方法一 可以看做是上面的ProxyFactoryBean和TransactionInterceptor合到一个类里去

假设我们要加事务的对象是

<bean id="userServiceTarget" class="sp.hibernate.UserService">

<property name="sessionFactory" ref="sessionFactory" />

直接将session工厂给它,通过getCurrentSession获得绑定在当前线程的session 直接通过session操作

如果还通过模板的话就显得有些鸡肋了

</bean>

接下来 创建一个事务管理器

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

<property name="sessionFactory" ref="sessionFactory" />

</bean>

再来是

本质就是一个ProxyFactorybean

<bean id=" userService" class="org.springframework.transaction.interceptor.TransactionProxyFactoryBean">

拦截的目标对象

<property name="target" ref=" userServiceTarget" />

事务管理器

<property name="transactionManager" ref="txManager" />

<!-- 这里采用类代理,因为我没有实现接口… -->

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

<!-- 事务属性 用来设置事务策略,即指定每个方法的事务策略(传播级别,只读…等) -->

<property name="transactionAttributes">

<props>

<!-- 要对add方法进行事务管理,可以使用通配符 -->

<prop key="add">PROPAGATION\_REQUIRED,ISOLATION\_DEFAULT</prop>

<prop key="find">PROPAGATION\_REQUIRED,ISOLATION\_DEFAULT,readOnly</prop>

<prop key="\*">PROPAGATION\_REQUIRED</prop> </props>

</property>

</bean>

可以看到这样真的是相当麻烦…(有一点优化,可以考虑来一个parent bean用于继承)

方法2

利用命名空间tx里的标签(方便了很多)

先来一个通知

<tx:advice id="txa" transaction-manager="txManager">

<tx:attributes>

<tx:method name="add" propagation="REQUIRED" isolation="DEFAULT" />

<tx:method name="find" propagation="REQUIRED" isolation="DEFAULT" read-only="true" />

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

</tx:attributes>

</tx:advice>

再来一个aop

<aop:config proxy-target-class="true">

//切入点

<aop:pointcut expression="execution (public \* sp.hibernate.UserService.\*(..) )" id="pc2" />

<aop:advisor advice-ref="txa" pointcut-ref="pc2" />

</aop:config>

然后我们的userService还是按照不同的方式定义就行了(省了很多配置)

方法3 注解

<!-- 注解自动包扫描 -->

<context:component-scan base-package="sp.hibernate2"/>

<!-- 启动注解 事务管理,并且指定事务管理器(如果名字就刚好是默认值就可以不指定 否则就要指定) -->

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

在我们的UserService上

@Service//用于spring

// 加载这里是给所有方法都加了事务管理

@Transactional(propagation = Propagation.REQUIRED, isolation = Isolation.READ\_COMMITTED)

//这里可以特化事务管理的配置

@Transactional(readOnly = **true**)

**public** Account find(**int** id) {

Session s = sessionFactory.getCurrentSession();

**return** (Account) s.get( Account.**class**, id );

}

注解也是非常方便

©