### Spring的声明式事物：

1. **在配置文件中配置事务管理器**
2. **启用事务注解**
3. **在要进行事务管理的方法上添加事务注解**

**<!-- 配置事务管理器 -->**

**<bean id=*"transactionManager"***

**class=*"org.springframework.jdbc.datasource.DataSourceTransactionManager"*>**

**<property name=*"dataSource"* ref=*"dataSource"*></property>**

**</bean>**

**<!-- 启用事务注解 -->**

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

//添加事务注解，说明该函数是一个事物

**@Transactional**

@Override

**public** **void** purchase(String username, String isbn) { //购买一本书，要更新书的库存和账户余额更新

//1.获取书的单价

**int** price = bookshopDao.findBookPriceByIsbn(isbn);

//2.更新书的库存

bookshopDao.updateBookStock(isbn);

//3.更新用户余额

bookshopDao.updateUserAccount(username, price);

}

**事务的传播行为：**

当一个事务被另一个事务方法调用时，必须制定事务该如何传播。例如：方法可能继续在现有的事务中运行，也可能开启一个新事务，在自己的事务中运行。

事务的传播行为可以由传播属性执行，spring定义了7种类型的传播行为，最常用的是REQUIRED和REQUIREDS\_NEW。

**事务的其他属性：**

范例：

**package** com.wp.tx;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Service;

**import** org.springframework.transaction.annotation.Isolation;

**import** org.springframework.transaction.annotation.Propagation;

**import** org.springframework.transaction.annotation.Transactional;

@Service("bookShopServiceImpl")

**public** **class** BookShopServiceImpl **implements** BookShopService{ //服务接口的实际操作类

@Autowired

**private** BookShopDao bookshopDao;

/\*\*

添加事务注解，说明该函数是一个事物

事务的五个属性：

**1.使用propagation指定事务的传播行为，即当前的事务方法被另外一个事务方法调用时如何使用**

**事务，默认取值为REQUIRED ，即使用调用方法的事务**

**2.REQUIREDS\_NEW：事务自己的事务，调用的事务方法的事务被挂起**

**3.使用isolation指定事务的隔离级别，最常用的取值为READ\_COMMITTED**

**默认情况下spring的声明式事务对所有的运行时异常进行回滚，也可以通过对应的属性进行设置。通常情况系取默认值**

**比如：noRollbackFor= {UserAccountException.class}对这个异常不进行回滚**

**4.使用readOnly指定事务是否为只读，表示这个事务只读取数据但是不更新数据，这样可以帮助数据库引擎优化事务**

**5.timeout属性：使用timeout指定强制回滚之前事务可以占用的时间，时间一到就强制回滚。**

\* \*/

/\* @Transactional(propagation=Propagation.REQUIRES\_NEW,

isolation=Isolation.READ\_COMMITTED,

noRollbackFor= {UserAccountException.class})\*/

@Transactional(propagation=Propagation.***REQUIRES\_NEW***,

isolation=Isolation.***READ\_COMMITTED***,

readOnly=**false**,

timeout=3)

@Override

**public** **void** purchase(String username, String isbn) { //购买一本书，要更新书的库存和账户余额更新

**try** {

Thread.*sleep*(2000); //线程睡2秒

} **catch** (InterruptedException e) {

e.printStackTrace();

}

//1.获取书的单价

**int** price = bookshopDao.findBookPriceByIsbn(isbn);

//2.更新书的库存

bookshopDao.updateBookStock(isbn);

//3.更新用户余额

bookshopDao.updateUserAccount(username, price);

}

}

**用注解来配置事务步骤：**

1. **在要进行事务回滚的方法前面添加事务注解（@Transactional）说明该方法是一个事务**
2. **在配置文件中配置事务管理器**
3. **启用事务注解**

范例：

<!-- 配置事务管理器 -->

<bean id=*"transactionManager"*

class=*"org.springframework.jdbc.datasource.DataSourceTransactionManager"*>

<property name=*"dataSource"* ref=*"dataSource"*></property>

</bean>

<!-- 启用事务注解 -->

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

**用xml文件形式来配置事务：**

**步骤：**

1. 在xml文件中正常的配置需要用到的bean
2. 配置事务管理器
3. 配置事务属性
4. 配置事务切入点，以及把切入点和事务属性关联起来

范例：

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:aop=*"http://www.springframework.org/schema/aop"*

xmlns:context=*"http://www.springframework.org/schema/context"*

xmlns:tx=*"http://www.springframework.org/schema/tx"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-2.0.xsd*

*http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.0.xsd*

*http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.0.xsd*

*http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.0.xsd"*>

**<!-- 设置自动扫描的包 -->**

<context:component-scan base-package=*"com.wp.tx"*></context:component-scan>

**<!-- 导入资源文件 -->**

<context:property-placeholder location=*"classpath:db.properties"* />

<**!-- 配置c3p0数据源 -->**

<bean id=*"dataSource"*

class=*"com.mchange.v2.c3p0.ComboPooledDataSource"*>

<property name=*"user"* value=*"${jdbc.user}"*></property>

<property name=*"password"* value=*"${jdbc.password}"*></property>

<property name=*"jdbcUrl"* value=*"${jdbc.jdbcUrl}"*></property>

<property name=*"driverClass"* value=*"${jdbc.driverClass}"*></property>

<property name=*"initialPoolSize"* value=*"${jdbc.initPoolSize}"*></property>

<property name=*"maxPoolSize"* value=*"${jdbc.maxPoolSize}"*></property>

</bean>

**<!-- 配置spring的JdbcTemplate -->**

<bean id=*"jdbcTemplate"*

class=*"org.springframework.jdbc.core.JdbcTemplate"*>

<property name=*"dataSource"* ref=*"dataSource"*></property>

</bean>

<!-- 配置 NamedParameterJdbcTemplate,该对象可以使用具名参数，其没有无参构造器，所以必须为其构造器指定参数-->

<bean id=*"NamedParameterJdbcTemplate"*

class=*"org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate"*>

<constructor-arg ref=*"dataSource"*></constructor-arg>

</bean>

**<!-- 配置bean -->**

<bean id=*"bookShopDao"* class=*"com.wp.tx.xml.BookShopDaoImpl"*>

<property name=*"jdbctemplate"* ref=*"jdbcTemplate"*></property>

</bean>

<bean id=*"bookShopService"* class=*"com.wp.tx.xml.service.impl.BookShopServiceImpl"*>

<property name=*"bookshopDao"* ref=*"bookShopDao"*></property>

</bean>

<bean id=*"cashier"* class=*"com.wp.tx.xml.service.impl.CashierImpl"*>

<property name=*"bookShopService"* ref=*"bookShopService"*></property>

</bean>

**<!-- 1.配置事务管理器 -->**

<bean id=*"transactionManager"*

class=*"org.springframework.jdbc.datasource.DataSourceTransactionManager"*>

<property name=*"dataSource"* ref=*"dataSource"*></property>

</bean>

**<!-- 2.配置事务属性 -->**

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

<tx:attributes>

<!-- 配置事务传播行为的属性，新开一个事务 -->

<tx:method name=*"purchase"* propagation=*"REQUIRES\_NEW"*/>

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

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

<tx:method name=*"\*"*/>

</tx:attributes>

</tx:advice>

**<!-- 3.配置事务切入点，以及把事务切入点和事务属性关联起来 -->**

<aop:config>

<aop:pointcut expression=*"execution(\* com.wp.tx.xml.service.\*.\*(..))"*

id=*"txPointCut"*/>

<!-- 将切入点和事务进行关联 -->

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

</aop:config>

</beans>

**五个事务属性：**

1. **事务隔离**
2. **事务的传播行为**
3. **事务的只读属性**
4. **事务的超时时间**
5. **事务的回滚规则**

**所谓事务管理，其实就是“按照给定的事务规则来执行提交或者回滚操作”**

**Spring 的声明式事务管理在底层是建立在 AOP 的基础之上的。其本质是对方法前后进行拦截，然后在目标方法开始之前创建或者加入一个事务，在执行完目标方法之后根据执行情况提交或者回滚事务。**

**虽然 @Transactional 注解可以作用于接口、接口方法、类以及类方法上，但是 Spring 小组建议不要在接口或者接口方法上使用该注解，因为这只有在使用基于接口的代理时它才会生效。另外， @Transactional 注解应该只被应用到 public 方法上，这是由 Spring AOP 的本质决定的。如果你在 protected、private 或者默认可见性的方法上使用 @Transactional 注解，这将被忽略，也不会抛出任何异常。**