@Transactional注解使用方法

# @Transactional注解的属性

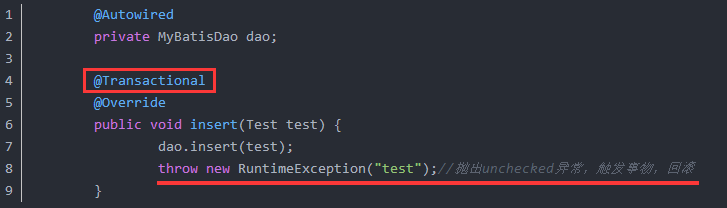
| **属性** | **类型** | **描述** |
| --- | --- | --- |
| **value** | String | 可选的限定描述符，指定使用的事务管理器 |
| **propagation** | enum: Propagation | 可选的事务传播行为设置 |
| **isolation** | enum: Isolation | 可选的事务隔离级别设置 |
| **readOnly** | boolean | 读写或只读事务，默认读写 |
| **timeout** | int (in seconds granularity) | 事务超时时间设置 |
| **rollbackFor** | Class对象数组，必须继承自Throwable | 导致事务回滚的异常类数组 |
| **rollbackForClassName** | 类名数组，必须继承自Throwable | 导致事务回滚的异常类名字数组 |
| **noRollbackFor** | Class对象数组，必须继承自Throwable | 不会导致事务回滚的异常类数组 |
| **noRollbackForClassName** | 类名数组，必须继承自Throwable | 不会导致事务回滚的异常类名字数组 |

# 用法

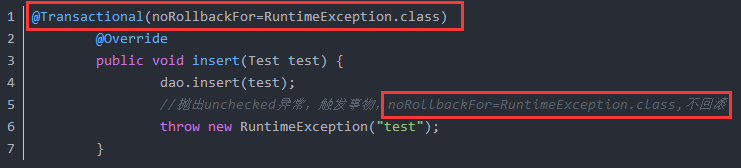
**@Transactional** 可以作用于**接口、接口方法**、**类以及类方法**上。当作用于类上时，该类的所有 public 方法将都具有该类型的事务属性，同时，我们也可以**在方法级别使用该标注来覆盖类级别的定义**。

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 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。

默认情况下，只有来自外部的方法调用才会被**AOP代理**捕获，也就是，类内部方法调用本类内部的其他方法并不会引起事务行为，即使被调用方法使用@Transactional注解进行修饰。



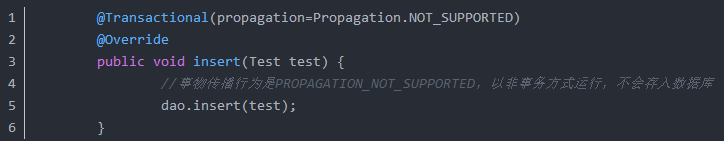
noRollbackFor=RuntimeException.class 运行时异常不回滚



**@Transactional**当作用于类上时，该类的所有 **public 方法**将都具有该类型的**事务属性**。



propagation=Propagation.NOT\_SUPPORTED



# myBatis为例 基于注解的声明式事务管理配置,xml配置



<!-- 事物切面配置 -->

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

**<tx:attributes>**

**<tx:method name="update\*" propagation="REQUIRED" read-only="false" rollback-for="java.lang.Exception"/>**

**<tx:method name="insert" propagation="REQUIRED" read-only="false"/>**

**</tx:attributes>**

**</tx:advice>**

**<aop:config>**

**<aop:pointcut id="testService" expression="execution (\* com.baobao.service.MyBatisService.\*(..))"/>**

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

**</aop:config>**