Spring事物传播属性

1、Spring传播属性包括：

（1）PROPAGATION\_REQUIRED --支持当期事务，如果当前没有事务，就新建一个事务。一般选择这个。

（2）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。

（3）PROPAGATION\_MANDATORY -- 支持当前事务，如果当前没有事务，就抛出异常。

（4）PROPAGATION\_REQUIRES\_NEW -- 新建事务，如果当前存在事务，把当前事务挂起。

（5）PROPAGATION\_NOT\_SUPPORTED -- 以非事务方式执行，如果当前存在事务，就把当前事务挂起。

（6）PROPAGATION\_NEBER -- 以非事务方式执行，如果当前存在事务，则抛出异常。

（7）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。

2、在Spring事务切面配置中遇到的问题。

在订单系统中，出现了先save后select操作，但是select操作查询到的数据竟然是save前的数据，导致业务异常。

其中，TransactionConfig.java中配置

*// 只读事务,不做更新操作*

RuleBasedTransactionAttribute readOnlyTx = **new** RuleBasedTransactionAttribute();

*// 设置只读属性*

readOnlyTx.setReadOnly(**true**);

*// 设置事务传播属性*

readOnlyTx.setPropagationBehavior(TransactionDefinition.***PROPAGATION\_NOT\_SUPPORTED***);

设置事物传播属性时，只读属性的事物传播属性设置了 ***PROPAGATION\_NOT\_SUPPORTED***

导致如上问题出现。将***PROPAGATION\_NOT\_SUPPORTED*** 修改为***PROPAGATION\_REQUIRED***，问题解决。

事务配置完整代码如下：

**import** org.aspectj.lang.annotation.Aspect;

**import** org.springframework.aop.Advisor;

**import** org.springframework.aop.aspectj.AspectJExpressionPointcut;

**import** org.springframework.aop.support.DefaultPointcutAdvisor;

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

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.Configuration;

**import** org.springframework.transaction.PlatformTransactionManager;

**import** org.springframework.transaction.TransactionDefinition;

**import** org.springframework.transaction.interceptor.\*;

**import** java.util.Collections;

**import** java.util.HashMap;

*/\*\**

*\* <p>ClassName:OMS </p>*

*\* <p>Description: 事务配置类</p>*

*\* <p>Company: http://www.shopin.net</p>*

*\** ***@author*** *zhangyong@shopin.cn*

*\** ***@version*** *1.0.0*

*\** ***@Date*** *2018/7/3 12:20*

*\*/*

@Aspect

@Configuration

**public class** TransactionConfig {

*//* 设置超时时间

*// private static final int TX\_METHOD\_TIMEOUT = 30;*

**private static final** String ***AOP\_POINTCUT\_EXPRESSION*** = **"execution (\* net.shopin.service.impl.\*.\*(..))"**;

@Autowired

**private** PlatformTransactionManager **transactionManager**;

@Bean

**public** TransactionInterceptor txAdvice() {

*// 可以实现对目标对象的每个方法实施不同的事务管理*

NameMatchTransactionAttributeSource source = **new** NameMatchTransactionAttributeSource();

*// 只读事务,不做更新操作*

RuleBasedTransactionAttribute readOnlyTx = **new** RuleBasedTransactionAttribute();

*// 设置只读属性*

readOnlyTx.setReadOnly(**true**);

*// 设置事务控制*

readOnlyTx.setPropagationBehavior(TransactionDefinition.***PROPAGATION\_REQUIRED***);

*// 当前存在事务就用当前事务,当前不存在事务就创建一个新的事务*

RuleBasedTransactionAttribute requiredTx = **new** RuleBasedTransactionAttribute();

*// 设置回滚规则*

requiredTx.setRollbackRules(Collections.*singletonList*(**new** RollbackRuleAttribute(Exception.**class**)));

requiredTx.setPropagationBehavior(TransactionDefinition.***PROPAGATION\_REQUIRED***);

*// requiredTx.setTimeout(TX\_METHOD\_TIMEOUT);*

HashMap<String, TransactionAttribute> txMap = **new** HashMap<>();

txMap.put(**"get\*"**, readOnlyTx);

txMap.put(**"select\*"**, readOnlyTx);

txMap.put(**"save\*"**, requiredTx);

txMap.put(**"create\*"**, requiredTx);

txMap.put(**"modify\*"**, requiredTx);

txMap.put(**"update\*"**, requiredTx);

txMap.put(**"cancel\*"**, requiredTx);

txMap.put(**"cancle\*"**, requiredTx);

txMap.put(**"delete\*"**, requiredTx);

txMap.put(**"merge\*"**, requiredTx);

txMap.put(**"remove\*"**, requiredTx);

txMap.put(**"overtime\*"**, requiredTx);

txMap.put(**"manualCancle\*"**, requiredTx);

txMap.put(**"des\*"**, requiredTx);

txMap.put(**"reg\*"**, requiredTx);

txMap.put(**"confirm\*"**, requiredTx);

txMap.put(**"pay\*"**, requiredTx);

txMap.put(**"repair\*"**, requiredTx);

source.setNameMap(txMap);

TransactionInterceptor txAdvice = **new** TransactionInterceptor(**transactionManager**, source);

**return** txAdvice;

}

@Bean

**public** Advisor txAdviceAdvisor() {

AspectJExpressionPointcut pointcut = **new** AspectJExpressionPointcut();

pointcut.setExpression(***AOP\_POINTCUT\_EXPRESSION***);

**return new** DefaultPointcutAdvisor(pointcut, txAdvice());

}

}