spring中基于XML的声明式事务控制配置步骤

一、配置

1、配置事务管理器

2、配置事务的通知

此时我们需要导入事务的约束 tx名称空间和约束，同时也需要aop的

使用tx:advice标签配置事务通知

属性：

id：给事务通知起一个唯一标识

transaction-manager：给事务通知提供一个事务管理器引用

3、配置AOP中的通用切入点表达式

4、建立事务通知和切入点表达式的对应关系

5、配置事务的属性

是在事务的通知tx:advice标签的内部

```
<!-- 配置事务管理器 -->
<bean id="transactionManager"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
   <property name="dataSource" ref="dataSource"></property>
</bean>

<!-- 配置事务的通知-->
<tx:advice id="txAdvice" transaction-manager="transactionManager">
   <!-- 配置事务的属性
        isolation：用于指定事务的隔离级别。默认值是DEFAULT，表示使用数据库的默认隔离级别。
        propagation：用于指定事务的传播行为。默认值是REQUIRED，表示一定会有事务，增删改的选择。查询方法可以选择SUPPORTS。
        read-only：用于指定事务是否只读。只有查询方法才能设置为true。默认值是false，表示读写。
        timeout：用于指定事务的超时时间，默认值是-1，表示永不超时。如果指定了数值，以秒为单位。
        rollback-for：用于指定一个异常，当产生该异常时，事务回滚，产生其他异常时，
```

事务不回滚。没有默认值。表示任何异常都回滚。
no-rollback-for：用于指定一个异常，当产生该异常时，事务不回滚，产生其他异常时事务回滚。没有默认值。表示任何异常都回滚。
-->
<tx:attributes>
<tx:method name="*" propagation="REQUIRED" read-only="false"/>
<tx:method name="find*" propagation="SUPPORTS" read-only="true">
</tx:method>
</tx:attributes>
</tx:advice>

<!-- 配置aop-->
<aop:config>
<!-- 配置切入点表达式-->
<aop:pointcut id="pt1" expression="execution(* com.itheima.service.impl.*.*
(..))"></aop:pointcut>
<!--建立切入点表达式和事务通知的对应关系 -->
<aop:advisor advice-ref="txAdvice" pointcut-ref="pt1"></aop:advisor>
</aop:config>

<?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:tx="http://www.springframework.org/schema/tx"
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/spring-tx.xsd
http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop.xsd">

<!-- 配置业务层-->
<bean id="accountService"
class="com.itheima.service.impl.AccountServiceImpl">
<property name="accountDao" ref="accountDao"></property>
</bean>

<!-- 配置账户的持久层-->
<bean id="accountDao" class="com.itheima.dao.impl.AccountDaoImpl">
<property name="dataSource" ref="dataSource"></property>
</bean>

<!-- 配置数据源-->

```
    <bean id="dataSource"
class="org.springframework.jdbc.datasource.DriverManagerDataSource">
        <property name="driverClassName" value="com.mysql.jdbc.Driver">
</property>
        <property name="url" value="jdbc:mysql://localhost:3306/eesy"></property>
        <property name="username" value="root"></property>
        <property name="password" value="1234"></property>
    </bean>
```

二、实例

执行transfer相当于一次事务，一旦发生异常，则回滚

```
@Override
public void transfer(String sourceName, String targetName, Float money) {
    System.out.println("transfer....");
        //2.1根据名称查询转出账户
        Account source = accountDao.findAccountByName(sourceName);
        //2.2根据名称查询转入账户
        Account target = accountDao.findAccountByName(targetName);
        //2.3转出账户减钱
        source.setMoney(source.getMoney()-money);
        //2.4转入账户加钱
        target.setMoney(target.getMoney()+money);
        //2.5更新转出账户
        accountDao.updateAccount(source);

        int i=1/0;

        //2.6更新转入账户
        accountDao.updateAccount(target);
}
```