spring中基于注解 的声明式事务控制配置步骤(仍需xml)

一、spring中基于注解 的声明式事务控制配置步骤

1、配置事务管理器

2、开启spring对注解事务的支持

3、在需要事务支持的地方使用@Transactional注解

在基于xml配置的基础上

```
<?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"
       xmlns:context="http://www.springframework.org/schema/context"
       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
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context.xsd">

    <!-- 配置spring创建容器时要扫描的包-->
    <context:component-scan base-package="com.itheima"></context:component-
scan>

    <!-- 配置JdbcTemplate-->
    <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
        <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>

    <!-- spring中基于注解 的声明式事务控制配置步骤
        1、配置事务管理器
        2、开启spring对注解事务的支持
        3、在需要事务支持的地方使用@Transactional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>


    <!-- 开启spring对注解事务的支持-->
    <tx:annotation-driven transaction-manager="transactionManager">
</tx:annotation-driven>
</beans>
```

因为transfer方法需要用到增删改，所以要另外设置，如果有多个需要改动的方法，就比较麻烦，这时用基于xml配置比较容易

```
@Service("account1service")

    //把这个实现类中的所有方法，注解为只读型事务配置
@Transactional(propagation = Propagation.SUPPORTS,readOnly = true)
public class Account1ServiceImpl implements IAccount1Service {
    //声明一个持久层的接口类属性，方便内部调用持久层方法。赋值采用注解 IOC 配置类型注入
    @Autowired
    private IAccount1Dao account1Dao;

    //id查询账户方法，业务层传给持久层。
    public Account1 findById(int id) {
        return account1Dao.findById(id);
    }

    /***转账接口方法
     * @param sourceName    转出账户名称
     * @param targetName    转入账户名称
     * @param money         转账金钱
     */
```

```
    //其中把单独的转账，也就是需要用到增删改相关的方法，注解为读写型事务配置
    @Transactional(propagation = Propagation.REQUIRED,readOnly = false)
    public void transfer(String sourceName, String targetName, double money) {
        System.out.println("transfter.转账事务开始.....");
        //1.根据名称查询转出账户
        Account1 source = account1Dao.findByName(sourceName);
        //2.根据名称查询转入账户
        Account1 target = account1Dao.findByName(targetName);
        //3.转出账户减钱
        source.setMoney(source.getMoney()-money);
        //4.转入账户加钱
        target.setMoney(target.getMoney()+money);
        //5.更新转出账户
        account1Dao.modify(source);
        //更新转入账户
        account1Dao.modify(target);
    }
}
```