事务控制

事务是开发中必不可少的东西。使用JDBC时，通过connection进行事务控制；使用MyBatis时，通过SqlSession进行事务控制。事务控制方式会发生变化，Spring在这些技术基础上，提供了统一的控制事务的接口，

事务控制方式	解释
编程式事务控制	Spring提供了事务控制的类和方法，使用编程的方式对业务代码进行事务控制。代码耦合度较高，一般不使用
声明式事务控制	Spring将事务控制的代码封装，对外提供了xml和注解配置方式，达到解耦合的作用，推荐使用

Spring事务编程相关的类主要有如下三个

事务控制相关类	解释
平台事务管理器 PlatformTransactionManager	是一个接口标准，实现类都具备事务提交、回滚和获得事务对象的功能，不同持久层框架可能会有不同实现方案
事务定义 TransactionDefinition	封装事务的隔离级别、传播行为、过期时间等属性信息
事务状态 TransactionStatus	存储当前事务的状态信息，如果事务是否提交、是否回滚、是否有回滚点等

虽然编程式事务控制我们不学习，但是编程式事务控制对应的这些类我们需要了解一下，因为我们在通过配置的方式进行声明式事务控制时也会看到这些类的影子

通过一个转账的例子，说明Spring的事务控制。

搭建一个转账的环境，dao层一个转出钱的方法，一个转入钱的方法，service层一个转账业务方法，内部分别调用dao层转出钱和转入钱的方法，准备工作如下：

- 数据库准备一个账户表tb_account;
- dao层准备一个AccountMapper，包括incrMoney和decrMoney两个方法；
- service层准备一个transferMoney方法，分别调用incrMoney和decrMoney方法；
- 在applicationContext文件中进行Bean的管理配置；
- 测试正常转账与异常转账。

使用Spring进行事务控制，有如下思路，

- 通知类是Spring提供的，需要导入Spring事务相关的坐标
- 配置目标类AcountServiceImpl
- 使用advisor标签配置切面

1. xml配置

xml配置事务管理的流程如下，

- 配置平台事务管理器，注意，不同的框架或数据库，使用的平台事务管理器实现可能不同

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

- 配置Spring提供的Advice。<tx:method>配置不同方法的事务属性，name表示方法名称，*代表通配符，它还有其他属性，
    - isolation属性指定事务的隔离级别，事务并发存在的三大问题：脏读、不可重复读、幻读，可通过设置事务的隔离级别保证并发问题的出现，常用的是READ_COMMITTED和REPETABLE_READ，具体信息见下图
    - timeout属性设置超时时间，默认-1，无超时时间，单位是秒
    - read-only属性，是否只读，默认false
    - propagation属性，事务的传播行为，解决业务方法调用业务方法（事务嵌套问题），见下图

```
<!--配置spring提供好的advice-->
<tx:advice id="txAdvice" transaction-manager="transactionManager">
  <tx:attributes>
    <tx:method name="*"/>
  </tx:attributes>
</tx:advice>
```

isolation属性	解释
DEFAULT	默认隔离级别，取决于当前数据库隔离级别，例如MySQL默认隔离级别是REPEATABLE_READ
READ_UNCOMMITTED	A事务可以读取到B事务尚未提交的事务记录，不能解决任何并发问题，安全性最低，性能最高
READ_COMMITTED	A事务只能读取到其他事务已经提交的记录，不能读取到未提交的记录。可以解决脏读问题，但是不能解决不可重复读和幻读
REPEATABLE_READ	A事务多次从数据库读取某条记录结果一致，可以解决不可重复读，不可以解决幻读
SERIALIZABLE	串行化，可以解决任何并发问题，安全性最高，但是性能最低

propagation属性：设置事务的传播行为，主要解决是A方法调用B方法时，事务的传播方式问题的，例如：使用单方的事务，还是A和B都使用自己的事务等。事务的传播行为有如下七种属性值可配置

事务传播行为	解释
REQUIRED（默认值）	A调用B，B需要事务，如果A有事务B就加入A的事务中，如果A没有事务，B就自己创建一个事务
REQUIRED_NEW	A调用B，B需要新事务，如果A有事务就挂起，B自己创建一个新的事务
SUPPORTS	A调用B，B有无事务无所谓，A有事务就加入到A事务中，A无事务B就以非事务方式执行
NOT_SUPPORTS	A调用B，B以无事务方式执行，A如有事务则挂起
NEVER	A调用B，B以无事务方式执行，A如有事务则抛出异常
MANDATORY	A调用B，B要加入A的事务中，如果A无事务就抛出异常
NESTED	A调用B，B创建一个新事务，A有事务就作为嵌套事务存在，A没事务就以创建的新事务执行

- 配置事务增强的AOP

```
<!--事务增强的AOP-->
<aop:config>
  <!--切点-->
  <aop:pointcut id="txPointcut" expression="execution(*
com.example.service.impl.*.*(..))"/>
  <!--织入，配置spring提供好的-->
  <aop:advisor advice-ref="txAdvice" pointcut-ref="txPointcut"/>
</aop:config>
```

2. 注解配置

- 配置平台事务管理器

```
@Bean
public TransactionManager transactionManager(DataSource dataSource) {
  return new DataSourceTransactionManager(dataSource);
}
```

- 配置事务增强的切入点，@Transactional注解的属性与xml配置中<tx:method>的属性相同

```
@Override
@Transactional
public void transferMoney(String sourceAccount, String targetAccount, Integer
money) {
  accountMapper.decrementMoney(sourceAccount, money);
//        int i = 1 / 0;
  accountMapper.incrementMoney(targetAccount, money);
}
```

- 开启@Transactional注解扫描，配置类中添加注解@EnableTransactionManagement