一、事务概念

事务是逻辑上的一组操作，要么都执行，要么都不执行。

#### 事务特性（ACID）



1. 原子性： 事务是最小的执行单位，不允许分割。事务的原子性确保动作要么全部完成，要么完全不起作用；
2. 一致性： 执行事务前后，数据保持一致；
3. 隔离性： 并发访问数据库时，一个用户的事物不被其他事物所干扰，各并发事务之间数据库是独立的；
4. 持久性: 一个事务被提交之后。它对数据库中数据的改变是持久的，即使数据库发生故障也不应该对其有任何影响。

二、Spring事务管理接口介绍

#### Spring事务管理接口

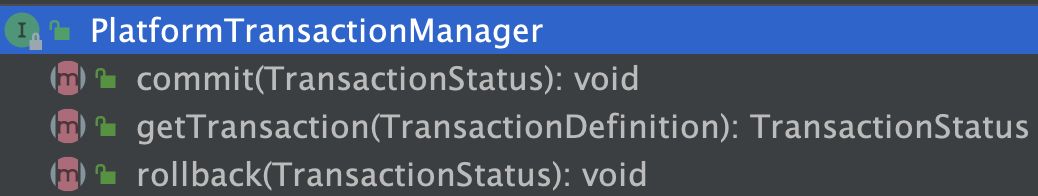
* PlatformTransactionManager： （平台）事务管理器
* TransactionDefinition： 事务定义信息(事务隔离级别、传播行为、超时、只读、回滚规则)
* TransactionStatus： 事务运行状态

所谓事务管理，其实就是“按照给定的事务规则来执行提交或者回滚操作”。

#### PlatformTransactionManager接口介绍

Spring并不直接管理事务，而是提供了多种事务管理器 ，他们将事务管理的职责委托给Hibernate或者JTA等持久化机制所提供的相关平台框架的事务来实现。 Spring事务管理器的接口是：

org.springframework.transaction.PlatformTransactionManager ，通过这个接口，Spring为各个平台如JDBC、Hibernate等都提供了对应的事务管理器，但是具体的实现就是各个平台自己的事情了。



Spring中的PlatformTransactionManager根据不同持久层框架所对应的接口实现类，几个常见的如下：



比如在使用JDBC或者iBatis进行数据持久化操作时，我们的xml配置通常放入下：

<!-- 事务管理器 -->

<bean id="transactionManager"

class="org.springframework.jdbc.datasource.DataSourceTransactionManager">

<!-- 数据源 -->

<property name="dataSource" ref="dataSource" />

</bean>

#### TansactionDefinition接口介绍

PlatformTransactionManager通过getTransaction(TransactionDefinition definition)方法聊得到一个事务，这个方法里的参数就是TransactionDefinition类，该类定义了一些基本的事务属性。

事务属性：可以理解成事务的一些基本配置，描述了事务策略如何应用到方法上。事务属性包含了5个方面。



TransactionDefinition接口中定义了5个方法以及一些表示事务属性的常量比如隔离级别、传播行为等等的常量。

// 返回事务的传播行为

int getPropagationBehavior();

// 返回事务的隔离级别，事务管理器根据它来控制另外一个事务可以看到本事务内的哪些数据

int getIsolationLevel();

// 返回事务必须在多少秒内完成

//返回事务的名字

String getName()；

int getTimeout();

// 返回是否优化为只读事务。

boolean isReadOnly();

1. 事务隔离级别（定义了一个事务可能受其他并发事务影响的程度）
2. 事务传播行为（为了解决业务层方法之间互相调用的事务问题）
3. 事务超时属性（一个事务允许执行的最长时间）
4. 事务只读属性（对事务资源是否执行只读操作）
5. 回滚规则（定义事务回滚规则）

#### TransactionStatus接口介绍

记录事务的状态。

PlatformTransactionManager.getTransaction(…) 方法返回一个 TransactionStatus 对象。返回的TransactionStatus 对象可能代表一个新的或已经存在的事务（如果在当前调用堆栈有一个符合条件的事务）。

接口方法如下：

boolean isNewTransaction(); // 是否是新的事物

boolean hasSavepoint(); // 是否有恢复点

void setRollbackOnly(); // 设置为只回滚

boolean isRollbackOnly(); // 是否为只回滚

boolean isCompleted; // 是否已完成

三、Spring支持两种方式的事务管理

#### 编程式事务管理（通过Transaction Template手动管理事务，实际应用中很少使用）

#### 使用XML配置声明式事务管理（实际通过AOP实现）

实现声明式事务的四种方式：

1. 基于 TransactionInterceptor 的声明式事务: Spring 声明式事务的基

础，通常也不建议使用这种方式，但是与前面一样，了解这种方式对理解

Spring 声明式事务有很大作用。

1. 基于 TransactionProxyFactoryBean 的声明式事务: 第一种方式的改进版本，简化的配置文件的书写，这是 Spring 早期推荐的声明式事务管理方式，但是在 Spring 2.0 中已经不推荐了。
2. 基于<tx> 和<aop>命名空间的声明式事务管理： 目前推荐的方式，其最大特点是与 Spring AOP 结合紧密，可以充分利用切点表达式的强大支持，使得管理事务更加灵活。
3. 基于 @Transactional 的全注解方式： 将声明式事务管理简化到了极致。开发人员只需在配置文件中加上一行启用相关后处理 Bean 的配置，然后在需要实施事务管理的方法或者类上使用 @Transactional 指定事务规则即可实现事务管理，而且功能也不必其他方式逊色。

#### 基于AspectJ的声明式事务管理

#### 基于注解的方式

<https://juejin.im/post/5b010f27518825426539ba38>