事务

真正管理事务的对象

org.springframework.jdbc.datasource.DataSourceTransactionManager 使用 SpringJDBC 或 或 iBatis 进行持久化数据时使用

org.springframework.orm.hibernate5.HibernateTransactionManager 使用 Hibernate

一、事务介绍：

1、例子：

你去ATM机取1000块钱，大体有两个步骤：首先输入密码金额，银行卡扣掉1000元钱；然后ATM出1000元钱。这两个步骤必须是要么都执行要么都不执行。

如果银行卡扣除了1000块但是ATM出钱失败的话，你将会损失1000元；

如果银行卡扣钱失败但是ATM却出了1000块，那么银行将损失1000元。

所以，如果一个步骤成功另一个步骤失败对双方都不是好事，如果不管哪一个步骤失败了以后，整个取钱过程都能回滚，也就是完全取消所有操作的话，这对双方都是极好的。

2、事务概念

事务是一系列的动作，一旦其中有一个动作出现错误，必须全部回滚，系统将事务中对数据库的所有已完成的操作全部撤消，滚回到事务开始的状态，避免出现由于数据不一致而导致的接下来一系列的错误。事务的出现是为了确保数据的完整性和一致性，在目前企业级应用开发中，事务管理是必不可少的。

```
获取连接 Connection con = DriverManager.getConnection()
开启事务con.setAutoCommit(true/false);
执行CRUD
提交事务/回滚事务 con.commit() / con.rollback();
关闭连接 conn.close();
```

二、四大特性

- **原子性（Atomicity）：事务是一个原子操作，由一系列动作组成。事务的原子性确保动作要么全部完成，要么完全不起作用。**

- **一致性（Consistency）：一旦事务完成（不管成功还是失败），系统必须确保它所建模的业务处于一致的状态，而不会是部分完成部分失败。在现实中的数据不应该被破坏。**
- **隔离性（Isolation）：可能有许多事务会同时处理相同的数据，因此每个事务都应该与其他事务隔离开来，防止数据损坏。**
- **持久性（Durability）：一旦事务完成，无论发生什么系统错误，它的结果都不应该受到影响，这样就能从任何系统崩溃中恢复过来。通常情况下，事务的结果被写到持久化存储器中。**

三、Spring 事务的传播属性

1、PROPAGATION_REQUIRED

支持当前事务，如果当前没有事务，就新建一个事务。这是最常见的选择，也是 Spring 默认的事务的传播。

2、PROPAGATION_REQUIRES_NEW

新建事务，如果当前存在事务，把当前事务挂起。新建的事务将和被挂起的事务没有任何关系，是两个独立的事务，外层事务失败回滚之后，不能回滚内层事务执行的结果，内层事务失败抛出异常，外层事务捕获，也可以不处理回滚操作

3、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。

4、PROPAGATION_MANDATORY

支持当前事务，如果当前没有事务，就抛出异常。

5、PROPAGATION_NOT_SUPPORTED

以非事务方式执行操作，如果当前存在事务，就把当前事务挂起。

6、PROPAGATION_NEVER

以非事务方式执行，如果当前存在事务，则抛出异常。

7、PROPAGATION_NESTED

如果一个活动的事务存在，则运行在一个嵌套的事务中。如果没有活动事务，则按REQUIRED属性执行。它使用了一个单独的事务，这个事务拥有多个可以回滚的保存点。内部事务的回滚不会对外部事务造成影响。它只对DataSourceTransactionManager事务管理器起效。

四、事务并发所可能存在的问题：

1.脏读：

一个事务读到另一个事务未提交的更新数据。也就是说，比如事务A的未提交（还依然缓存）的数据被事务B读走，如果事务A失败回滚，会导致事务B所读取的的数据是错误的。

2.不可重复读：

一个事务两次读同一行数据，可是这两次读到的数据不一样。比如事务A中两处读取数据-total-的值。在第一读的时候，total是100，然后事务B就把total的数据改成 200，事务A再读一次，结果就发现，total竟然就变成200了，造成事务A数据混乱。

3.幻读：

一个事务执行两次查询，但第二次查询比第一次查询多出了一些数据行。这个和non-repeatable reads相似，也是同一个事务中多次读不一致的问题。但是non-repeatable reads的不一致是因为他所要取的数据集被改变了（比如total的数据），但是phantom reads所要读的数据的不一致却不是他所要读的数据集改变，而是他的条件数据集改变。比如Select account.id where account.name="ppgogo*",第一次读去了6个符合条件的id，第二次读取的时候，由于事务b把一个帐号的名字由"dd"改成"ppgogo1"，结果取出来了7个数据。

4.丢失更新：

撤消一个事务时，把其它事务已提交的更新的数据覆盖了。

五、Spring中的隔离级别

1、ISOLATION_DEFAULT

这是个 PlatfromTransactionManager 默认的隔离级别，使用数据库默认的事务隔离级别。另外四个与 JDBC 的隔离级别相对应。

2、ISOLATION_READ_UNCOMMITTED

这是事务最低的隔离级别，它充许另外一个事务可以看到这个事务未提交的数据。这种隔离级别会产生脏读，不可重复读和幻像读。

3、ISOLATION_READ_COMMITTED

保证一个事务修改的数据提交后才能被另外一个事务读取。另外一个事务不能读取该事务未提交的数据。

4、ISOLATION_REPEATABLE_READ

这种事务隔离级别可以防止脏读，不可重复读。但是可能出现幻像读。

5、ISOLATION_SERIALIZABLE

这是花费最高代价但是最可靠的事务隔离级别。事务被处理为顺序执行。

六、PlatformTransactionManager

Spring事务管理的核心接口是PlatformTransactionManager

主要方法：

```
int getIsolationLevel();// 返回事务的隔离级别
String getName();// 返回事务的名称
int getPropagationBehavior();// 返回事务的传播行为
int getTimeout();  // 返回事务必须在多少秒内完成
boolean isReadOnly(); // 事务是否只读，事务管理器能够根据这个返回值进行优化，确保事务是只读的
```