事务是访问数据库的一个操作序列，数据库应用系统通过事务集来完成对数据可的存取。事务的正确执行使得数据库从一种状态转化为另一种状态

事务的特性：

原子性：事务中的所有操作，要么全部成功，要么全部失败

一致性：在事务开始与结束时，数据库处于一致状态

隔离性：多个事务并发执行，一个事务的执行不影响其他事务的执行

持久性：在事务结束时，此操作将不可逆转

并发下事务产生的问题：

脏读：事务A读取到了事务B还未提交的数据。例如银行取钱，事务A开启事务，此时切换到事务B，事务B开启事务🡪取走100元，此时切换回事务A，事务A读取的肯定是数据库的原始数据，因为事务B取走100块钱，还没提交，数据库里面的账务余额肯定还是原始数据，这就是脏读。

不可重复读：在一个事务里面读取了两次某个数据，读出来的数据不一致。还是以银行取钱为例，事务A开启事务🡪查出银行卡余额为1000元，此时切换到事务B，事务B开启事务🡪取走100元🡪提交，数据库里面余额变成900元，此时切换回事务A，事务A再查一次查出账户余额为900，这样对事务A而言，在同一个事务内两次读取账户余额数据不一致，这就是不可重复读。

幻读：在一个事务里面的操作中发现了未被操作的数据。比如学生信息，事务A开启事务🡪修改所有学生当天签到状况为false，此时切换到事务B，事务B开启事务🡪事务B插入了一条学生数据，此时切换回事务A，事务A提交的时候发现了一条自己没有修改过的数据，这就是幻读，就好像发生了幻觉一样。幻读出现的前提是并发的事务中有事务发生了插入、删除操作

事务隔离级别：事务隔离级别，就是为了解决并发事务产生的问题而诞生的。事务隔离级别越高，在并发下会产生的问题就越少，但同时付出的性能消耗也越大，因此很多时候必须在并发行和性能之间做一个权衡。

DEFAULT:默认隔离级别，每种数据库支持的事务隔离级别不一样，如果spring配置事务时将isolation设置为这个值的话，那么将使用底层数据库的默认事务隔离级别。

READ\_UNCOMMITTED：读未提交，即能够读取到没有提交的数据，所有很明显这个级别的隔离机制无法解决脏读、不可重复度、幻读中的任何一种。

READ\_COMMITTTED：读已提交，即能够读到那些已经提交的数据，自然能够防止脏读，但无法限制不可重复读和幻读。

REPEATABLE\_READ：重复读取，即在数据读出来之后加锁，明确数据读取出来就是为了更新用的，所有要加一把锁，防止别人修改它，读取了一条数据，这个事务不结束，别的事务就不可以改这条数据，这样就解决了脏读、不可重复读的问题，但无法解决幻读的问题。

SERIALIZABLE:串行化，最高的事务隔离级别，不管多少事务，挨个运行完一个事务的所有子事务之后才可以执行另一个事务里面的所有子事务，这样就解决了脏读、不可重复读、幻读的问题

隔离级别 脏读 不可重复读 幻读 加锁读

READ\_UNCOMMITTED 是 是 是 否

READ\_COMMITTED 否 是 是 否

REPEATABL\_READ 否 否 是 否

SERIALIZABLE 否 否 否 是

事务的传播机制：

Spring的默认传播机制是REQUIRED

嵌套事务：如果外部事务回滚，则嵌套事务也会回滚；嵌套事务回滚不会影响外部事务；外部事务提交时，嵌套事务才会提交

REQUIRED:当前如果有事务，使用该事务；否则会开始一个新事务

SUPPORTS:当前如果有事务，使用该事务；否则不会开始一个新事务

MANDATORY：当前如果有事务，使用该事务；否则抛出异常

REQUIRES\_NEW：总是开启一个新事；如果当前有事务，则该事务挂起

NOT\_SUPPORTS：以非事务的方式执行；如果当前有事务，则该事务挂起

NEVER：以非事务的方式执行；如果当前有事务，则抛出异常

NESTED：如果当前有事务，则嵌套在事务中执行；否则会新开始一个事务

多数据源事务：

配置多数据源时，需要设置一个主数据源，多数据源Transactional默认使用主数据源的事务管理机制，可以同过设置value来切换事务管理机制。多数据源在同一个事务里面时，Transactional只会回滚所配置的事务管理机制的数据源的事务