Spring事务的隔离级别和传播特性

事务传播机制：事务的传播性一般用在事务嵌套的场景，比如一个事务方法里面调用了另外一个事务方法，那么两个方法是各自作为独立的方法提交还是内层的事务合并到外层的事务一起提交，这就需要事务传播机制的配置来确定怎么样执行；在TransactionDefinition接口中定义了以下几个表示传播机制的常量，值为0~6：

TransactionDefinition接口定义以下特性：

1. TransactionDefinition.PROPAGATION_REQUIRED：默认值，能满足绝大部分业务需求，如果外层有事务，则当前事务加入到外层事务，一块提交，一块回滚。如果外层没有事务，新建一个事务执行；
2. TransactionDefinition.PROPAGATION_REQUIRES_NEW：该事务传播机制是每次都会新开启一个事务，同时把外层事务挂起，直到当前事务执行完毕，恢复上层事务的执行。如果外层没有事务，执行当前新开启的事务即可；
3. TransactionDefinition.PROPAGATION_SUPPORTS：如果外层有事务，则加入外层事务；如果外层没有事务，则直接以非事务的方式继续运行。完全依赖外层的事务；
4. TransactionDefinition.PROPAGATION_NOT_SUPPORTED：该传播机制不支持事务，如果外层存在事务则挂起，执行完当前代码，则恢复外层事务，无论是否异常都不会回滚当前的代码；
5. TransactionDefinition.PROPAGATION_NEVER：该传播机制不支持外层事务，即如果外层有事务就抛出异常并回滚事务；
6. TransactionDefinition.PROPAGATION_MANDATORY：与NEVER相反，如果外层有事务，则加入外层事务，如果外层没有事务，则抛出异常；
7. TransactionDefinition.PROPAGATION_NESTED：该传播机制的特点是可以保存状态保存点，当前事务回滚到某一个点，从而避免所有的嵌套事务都回滚，即各自回滚各自的，嵌套事务可以独立于当前事务进行单独地提交或回滚，如果子事务没有把异常吃掉，基本还是会引起全部回滚的。如果当前存在事务，则在嵌套事务内执行。如果当前没有事务，则执行与REQUIRED类似的操作。无论外层是否有事务，都只作为一次事务提交；

事务隔离级别：指若干个并发的事务之间的隔离程度，TransactionDefinition接口中定义了5个表示隔离级别的常量

1. TransactionDefinition.ISOLATION_DEFAULT：默认值-1，表示使用底层数据库的默认隔离级别，对大部分数据库而言，通常这值就是TransactionDefinition.ISOLATION_READ_COMMITTED；

2. TransactionDefinition.ISOLATION_READ_UNCOMMITTED：该隔离级别表示一个事务可以读取另一个事务修改但还没有提交的数据，该级别可能导致脏读、不可重复读和幻读，因此很少使用该隔离级别，比如PostgreSQL实际上并没有此级别；
3. TransactionDefinition.ISOLATION_READ_COMMITTED：（Oracle默认级别）该隔离级别表示一个事务只能读取另一个事务已经提交的数据，即允许从已经提交的并发事务读取，该级别可以防止脏读，但幻读和不可重复读仍可能会发生；
4. TransactionDefinition.ISOLATION_REPEATABLE_READ：（MySQL默认级别）该隔离级别表示一个事务在整个过程中可以多次重复执行某个查询，并且每次返回的记录都相同，即对相同字段的多次读取的结果是一致的，除非数据被当前事务本事改变。该级别可以防止脏读和不可重复读，但幻读仍可能发生；
5. TransactionDefinition.ISOLATION_SERIALIZABLE：（完全服从ACID的隔离级别）所有的事务依次逐个执行，这样事务之间就完全不可能产生干扰，也就是说，该级别可以防止脏读、不可重复读和幻读，但严重影响程序的性能，因为它通常是通过完全锁定当前事务所涉及的数据表来完成的；

Spring事务回滚规则：默认配置下，Spring只有在抛出的异常为运行时异常（runtime exception）时才回滚该事务，也就是抛出的异常为RuntimeException的子类（Error也会导致事务回滚），而抛出受检查异常（checked exception）则不会导致事务回滚，不过可以声明在抛出哪些异常时回滚事务，包括checked异常，也可以声明哪些异常抛出时不回滚事务，即使异常是运行时异常，还可以编程性的通过setRollbackOnly()方法来指示一个事务必须回滚，在调用完setRollbackOnly()后你所能执行的唯一操作就是回滚；