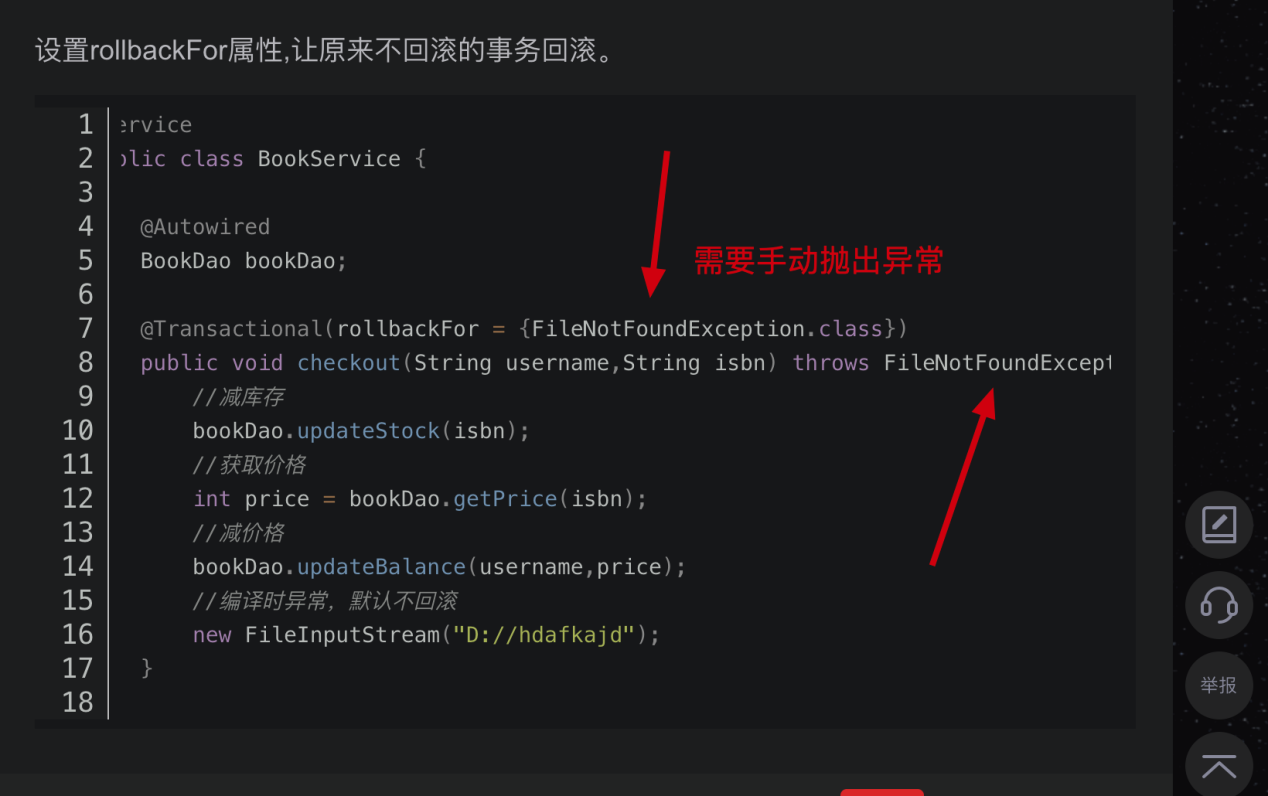
**分布式事务**

普通事务

异常分类：  
运行时异常：可以不处理，默认都回滚  
编译器异常：必须处理，默认不回滚

就是在方法后面抛出的异常需要手动处理



参考博客：

<https://blog.csdn.net/qq_42764468/article/details/103321663>

<https://my.oschina.net/happyBKs/blog/513441>

<https://blog.csdn.net/lihuarongaini/article/details/101299527>

isolation隔离级别

参数讲解：

//添加事务注解

//使用propagation指定事务的传播行为，即当前事务的方法被另一个事务方法调用时，如何使用是事务。

//使用另外一个方法的事务（REQUIRED）？还是另开一个新事物？（REQUIRES\_NEW）

//默认REQUIRED

//1. REQUIRES\_NEW:事务自己的事务，调用该事物方法的事务被挂起

//

//2. 用propagation指定事务的隔离级别，最常用的是READ\_COMMITTED

//3. 使用readOnly指定事务是否为只读，表示这个事务只读取数据，不更新数据，这样可以帮助数据库引擎优化事务。

//如果真的是一个只读取数据库值的方法，应该设置readOnly=true

//4. 使用timeout设置一个时间限额，单位为秒，如果事务的执行时间超过这个时间阈值，事务将被强制回滚。

参考视频：<https://www.bilibili.com/video/BV1d741137pc?from=search&seid=11435031562743319803>

seata框架

<https://blog.csdn.net/zhangcongyi420/article/details/104729699/>

# 分布式事务协议

2pc、3pc

参考博客：<https://www.cnblogs.com/monkeyblog/p/10449363.html>

## 2pc:

|  |
| --- |
| **1、二阶段提交方案：强一致性**  事务的发起者称协调者，事务的执行者称参与者。  处理流程：  　　1、准备阶段  　　　　事务协调者，向所有事务参与者发送事务内容，询问是否可以提交事务，并等待参与者回复。  　　　　事务参与者收到事务内容，开始执行事务操作，讲 undo 和 redo  信息记入事务日志中（但此时并不提交事务）。  　　　　如果参与者执行成功，给协调者回复yes,表示可以进行事务提交。如果执行失败，给协调者回复no,表示不可提交。  　　2、提交阶段  　　　　如果协调者收到了参与者的失败信息或超时信息，直接给所有参与者发送回滚（rollback）信息进行事务回滚，否则，发送提交（commit）信息。  　　　　参与者根据协调者的指令执行提交或者回滚操作，释放所有事务处理过程中使用的锁资源。(注意：必须在最后阶段释放锁资源) 接下来分两种情况分别讨论提交阶段的过程。 |

## 3pc

# 分布式事务解决方案

seate文档: <http://seata.io/zh-cn/docs/overview/faq.html>