# 参考

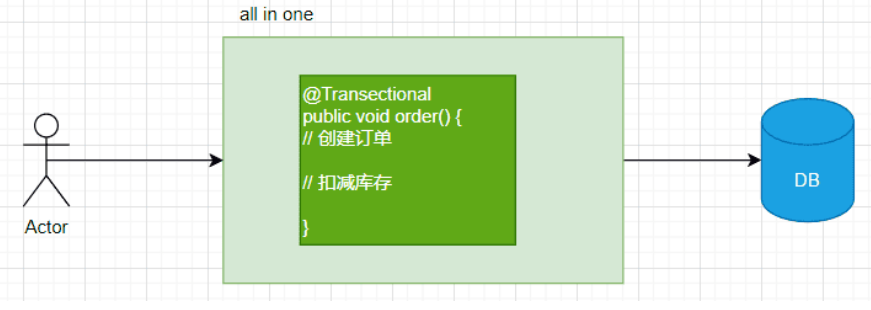
正本清源-分布式事务之Seata

https://pdai.tech/md/arch/arch-z-transection.html

# 理论

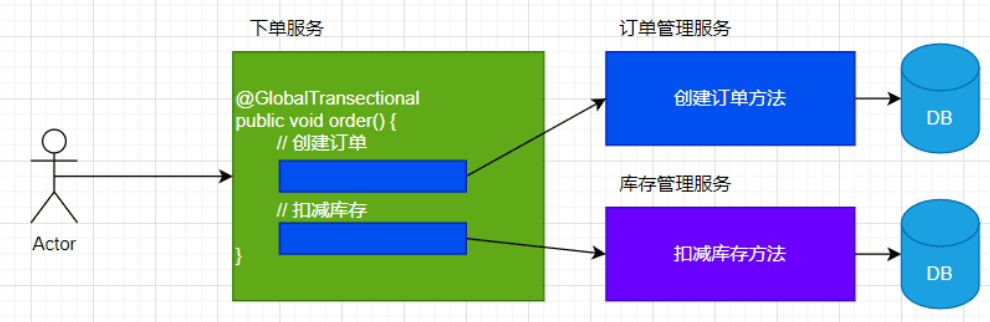
**场景：**下单减库存，涉及到订单创建、库存扣减

**单体场景：**使用同一个数据库，借助本地事务保证，如spring的@Transactional注解



**分布式场景**：独立的订单中心，库存中心，各自单独的数据库。创建订单和扣减库存，需要同时对订单DB和库存DB进行操作。两步操作必须同时成功，否则就会造成业务混乱，可此时我们只能保证自己服务的数据一致性，无法保证调用其他服务的操作是否成功，所以为了保证整个下单流程的数据一致性，就需要分布式事务介入。

如下GlobalTransactional注解采用阿里Seata的AT模式下的注解



## CAP/BASE

分布式的理论基础是CAP，由于P(分区容错）是必选项，所以只能在AP或者CP中选择。

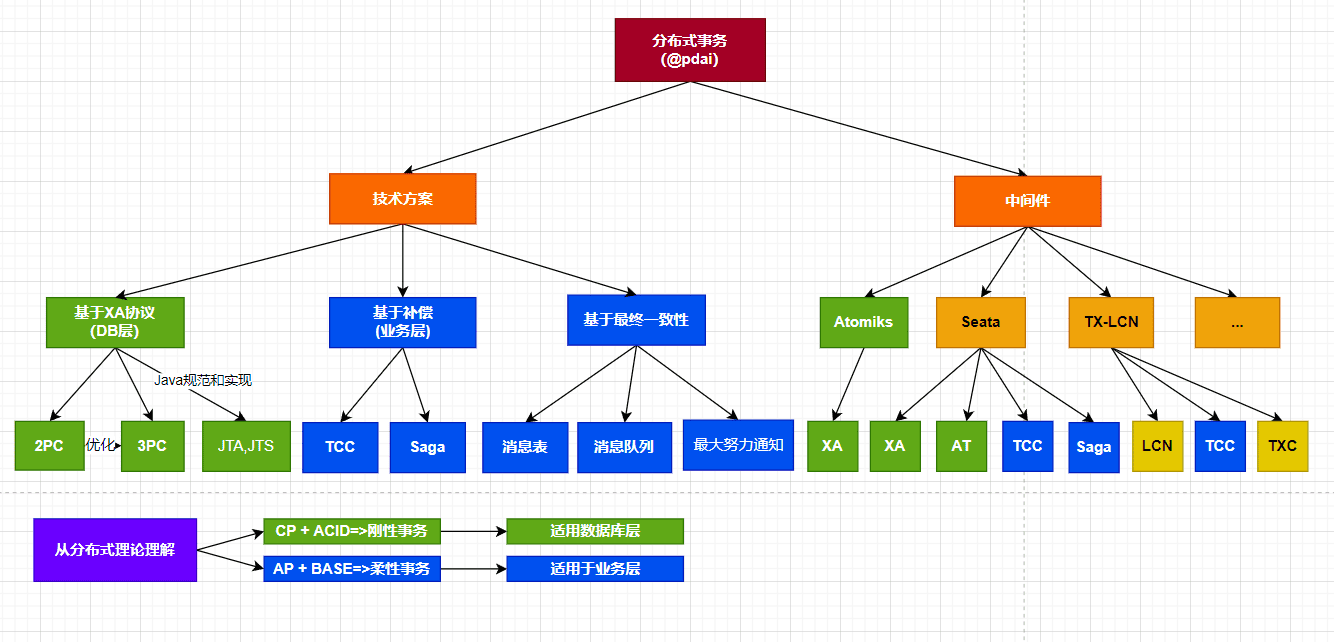
* **分布式理论的CP -> 刚性事务**

遵循ACID，对数据要求强一致性

* **分布式理论的AP+BASE -> 柔性事务**

遵循BASE，允许一定时间内不同节点的数据不一致，但要求最终一致

## 分布式事务知识体系



## 刚性事务

### 两阶段提交2PC

### 三阶段提交3PC

## 柔性事务

### TCC

### 本地消息表

### MQ事务方案（可靠消息事务）

### 最大努力通知

# 中间件Seata

### AT模式

### XA模式

### TCC模式

### Saga模式