

[**#**](https://pig4cloud.com/#%E4%BB%80%E4%B9%88%E6%98%AF%E5%88%86%E5%B8%83%E5%BC%8F%E4%BA%8B%E5%8A%A1) **什么是分布式事务**

上图所示，当请求A模块时

1. A模块Mapper直接操作DB 得到数据集
2. 使用Feign 调用 B模块
3. B模块Mapper操作DB 得到数据集
4. 合并1.3 两步操作或者其他的操作

[**#**](https://pig4cloud.com/#%E5%88%86%E6%9E%90%E4%B8%8A%E5%9B%BE%E5%8F%AF%E8%83%BD%E5%8F%91%E7%94%9F%E5%BC%82%E5%B8%B8%E5%8F%8A%E5%85%B6%E5%9B%9E%E6%BB%9A) **分析上图可能发生异常及其回滚**

* 步骤1 发生异常， spring 事务会自动回滚
* 步骤2 发生异常， spring 事务会自动回滚
* 步骤3 发生异常， 3会自动回滚， 2根据返回码抛出自定义异常 1会自动回滚
* 步骤4 发生异常， spring 事务会自动1 的操作 ，但是3的操作不会回滚

分布式事务的目标 当A模块开启事务时，如果4 操作过程异常，3的操作应该回滚

[**#**](https://pig4cloud.com/#pigx-%E4%B8%AD%E5%88%86%E5%B8%83%E5%BC%8F%E4%BA%8B%E5%8A%A1) **pigX 中分布式事务**

基于 LCN 4.1 深度定制，事务操作和代码耦合度极低，支持注册中心的事务发现和自动管理，用法完全兼容LCN 。以下操作均为简单演示、更负责请参考LCN 官网说明

* [LCN官网](https://www.txlcn.org/zh-cn/)

[**#**](https://pig4cloud.com/#a%2Fb-%E6%A8%A1%E5%9D%97%E5%BC%95%E5%85%A5-%E4%BA%8B%E5%8A%A1%E7%AE%A1%E7%90%86%E4%BE%9D%E8%B5%96) **A/B 模块引入 事务管理依赖**

<!--工具类核心包-->

<dependency>

<groupId>com.pig4cloud</groupId>

<artifactId>pigx-common-transaction</artifactId>

</dependency>

[**#**](https://pig4cloud.com/#a%E6%A8%A1%E5%9D%97%E7%9A%84service-%E5%BC%80%E5%90%AF%E4%BA%8B%E5%8A%A1) **A模块的Service 开启事务**

* 注意这里A作为Feign 的发起方 这里isStart = true 来声明是事务的发起方

@TxTransaction(isStart = true)

@Transactional(rollbackFor = Exception.class)

public Object save() {

}

[**#**](https://pig4cloud.com/#b%E6%A8%A1%E5%9D%97%E7%9A%84service-%E5%90%8C%E6%A0%B7%E5%BC%80%E5%90%AF%E4%BA%8B%E5%8A%A1) **B模块的Service 同样开启事务**

@TxTransaction

@Transactional(rollbackFor = Exception.class)

public Object save() {

}

[**#**](https://pig4cloud.com/#%E5%90%AF%E5%8A%A8tx-manager-%E4%BA%8B%E5%8A%A1%E5%8D%8F%E8%B0%83%E6%A8%A1%E5%9D%97) **启动tx-manager 事务协调模块**

PigxTxManagerApplication

* 注意， 先启动tx-manage 然后再启动你的业务模块

这样就完成了分布式事务的接入。