# 代理（Proxy）模式

作用：

通过代理，可以控制访问某个对象的方法，在调用这个方法前做前置处理，调用这个方法后做后置处理

分类：

静态代理(静态定义代理类)

动态代理(动态生成代理类)

JDK自带的动态代理(接口)

CGLIB

核心角色：

抽象角色(接口)

– 定义公共对外方法

目标对象

代理对象

代理模式的关键点是:代理对象与目标对象.代理对象是对目标对象的扩展,并会调用目标对象

静态代理的缺点：

代理对象需要与目标对象实现一样的接口,所以会有很多代理类,类太多.且大量代码冗余。同时,一旦接口增加方法,目标对象与代理对象都要维护.

动态代理：

Jdk:

代理类不用实现公共接口，实现InvocationHandler接口，在由invoke调用目标对象。

Cglib:

在没有接口的情况下使用

# Sping的AOP

aop：面向切面（方面）编程，扩展功能不修改源代码

aop：采用横向抽取机制，取代了传统纵向继承体系重复性代码

底层原理实现了jdk动态代理

## AOP操作相关术语：

1、连接点（joinpoint）

被拦截到的点，因为Spring只支持方法类型的连接点，所以在Spring中连接点指的就是被拦截到的方法。

**2、切入点（pointcut）**

切入点是指我们要对哪些连接点进行拦截的定义

**3、通知（advice）**

所谓通知指的就是指拦截到连接点之后要执行的代码，通知分为前置、后置、异常、最终、环绕通知五类

**4、切面（aspect）**

是切入点和通知的结合

### 使用表达式配置切入点：

**execution(<访问修饰符>? <方法名>(<参数>)<异常>)**

execution(\* com.usian.service.\*.\*(..))

# 事务

## 什么是事务？

当你需要一次执行多条SQL语句时，可以使用事务。通俗一点说，如果这几条SQL语句全部执行成功，则才对数据库进行一次更新，如果有一条SQL语句执行失败，则这几条SQL语句全部不进行执行，这个时候需要用到事务。

### 事务的四大特性：

原子性（Atomicity）

原子性是指事务是一个不可分割的工作单位，事务中的操作要么都执行，要么都不执行。

一致性（Consistency）

事务前后的数据都是正确的

隔离性（Isolation）

事物之间相互隔离，互不干扰（并发执行的事务彼此无法看到对方的中间状态）

事务的隔离性是多个用户并发访问数据库时，数据库为每一个用户开启的事务，不能被其他事务的操作数据所干扰，事务隔离可以避脏读等问题。

持久性（Durability）

持久性是指一个事务一旦被提交，该事务对数据的更改会被持久化到数据库，且不会被回滚。

## 分布式事务 什么是分布式事务

分布式系统会把一个应用系统拆分为可独立部署的多个服务，因此需要服务与服务之间远程协作才能完成事务操作，这种分布式系统环境下由不同的服务之间通过网络远程协作完成事务称之为**分布式事务**，例如用户注册送积分事务、创建订单减库存事务，银行转账事务等都是分布式事务。

## 分布式事物产生的场景

1. 典型的场景就是微服务架构 微服务之间通过远程调用完成事务操作。 比如：订单微服务和库存微服务，下单的同时订单微服务请求库存微服务减库存。简言之：跨JVM进程产生分布式事务。
2. 单体系统访问多个数据库实例 当单体系统需要访问多个数据库（实例）时就会产生分布式事务。 比如：用户信息和订单信息分别在两个MySQL实例存储，用户管理系统删除用户信息，需要分别删除用户信息及用户的订单信息，由于数据分布在不同的数据实例，需要通过不同的数据库链接去操作数据，此时产生分布式事务。 简言之：跨数据库实例产生分布式事务。

3、多服务访问同一个数据库实例 比如：订单微服务和库存微服务即使访问同一个数据库也会产生分布式事务，两个微服务持有了不同的数据库链接进行数据库操作，此时产生分布式事务。简言之：跨JVM进程产生分布式事务。

# RabbitMQ可靠消息最终一致性介绍

## 为什么要使用可靠消息最终一致性：

针对这种基于 MQ 的异步调用，如何保证各个服务间的分布式事务呢？也就是说，我希望的是基于MQ 实现异步调用的多个服务的业务逻辑，要么一起成功，要么一起失败。这个时候，就要用上可靠消息最终一致性方案，来实现分布式事务。

## 什么是可靠消息最终一致性：

可靠消息最终一致性方案是指当事务发起方执行完成本地事务后并发出一条消息，事务参与方(消息消费者)一定能够接收消息并处理事务成功，此方案强调的是只要消息发给事务参与方最终事务要达到一致。

## 可靠消息最终一致性要解决的问题：

### 上游服务把信息成功发送：

**本地事务与消息发送的原子性问题**：事务发起方在本地事务执行成功后消息必须发出去，否则就回滚事务。即实现本地事务和消息发送的原子性，要么都成功，要么都失败。

### 下游服务成把消息成功消费：

事务参与方接收消息的可靠性：事务参与方必须能够从消息队列接收到消息。

### 对消息做幂：

**消息重复消费的问题**：由于网络2的存在，若某一个消费节点响应超时但是消费成功，此时消息中间件会重复投递此消息，就导致了消息的重复消费。

# 解决方案

### **问题一：上游服务把消息成功发送**

针对问题一可采用消息表这个方案，该方案最初是eBay提出的，此方案的核心是：在系统A处理任务完成后，在本地记录待发送信息。一个定时任务不断检查，是否发送成功，如果发送成功，将记录状态修改。

### **问题二：下游服务成把消息成功消费**

**消息重试：**消息持久化后，如果消息在投递过程中丢失，或消息的确认应答在返回途中丢失，那么消息中间件就会重新投递，直到下游消费者返回消费成功响应为止。

**任务失败：**当任务处理失败后，则返回给消息中间件失败，消息会重复发送

### **问题三：对消息做幂等**

任务B处理消息前，先查询该消息是否被消费，如果没消费，处理任务B成功，记录消息。如果消息已经被消费，直接返回应答成功