1. MQ思路

本质：中间件 – 数据存储  
MQ区别： 开发语言 – （接收请求 – 存储 – 分发）

ActiveMQ --- java语言阵营，针对MQ这种中间件的全面实现 --

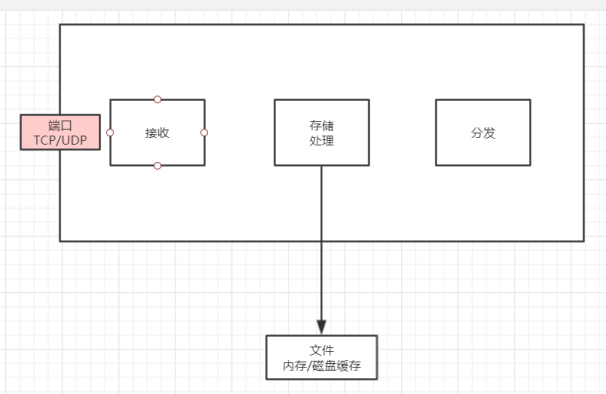
RabbitMQ – amqp – 性能/可靠/入手简单（spring / rabbitmq 同一家公司维护）

Kafka -- 海量数据传输（日志）

RocketMQ – 后发优势 – 总结前人的经验，再次进行设计（借鉴了很多中间件的设计）

持久化—刷盘 – 内存持久化到文件

性能的关键 – 持久化 < 非持久化性能 ---

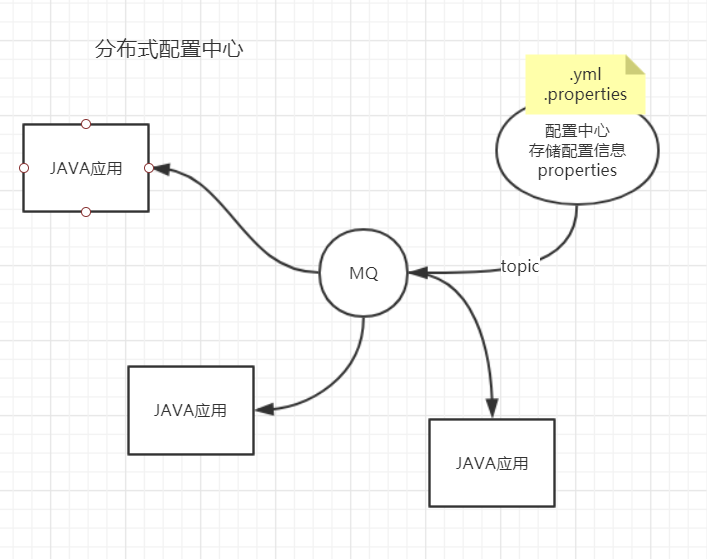


1. MQ应用

**有序消息**-场景 金额变动： 重点应该消费者端，不能多线程，多线程无法控制顺序

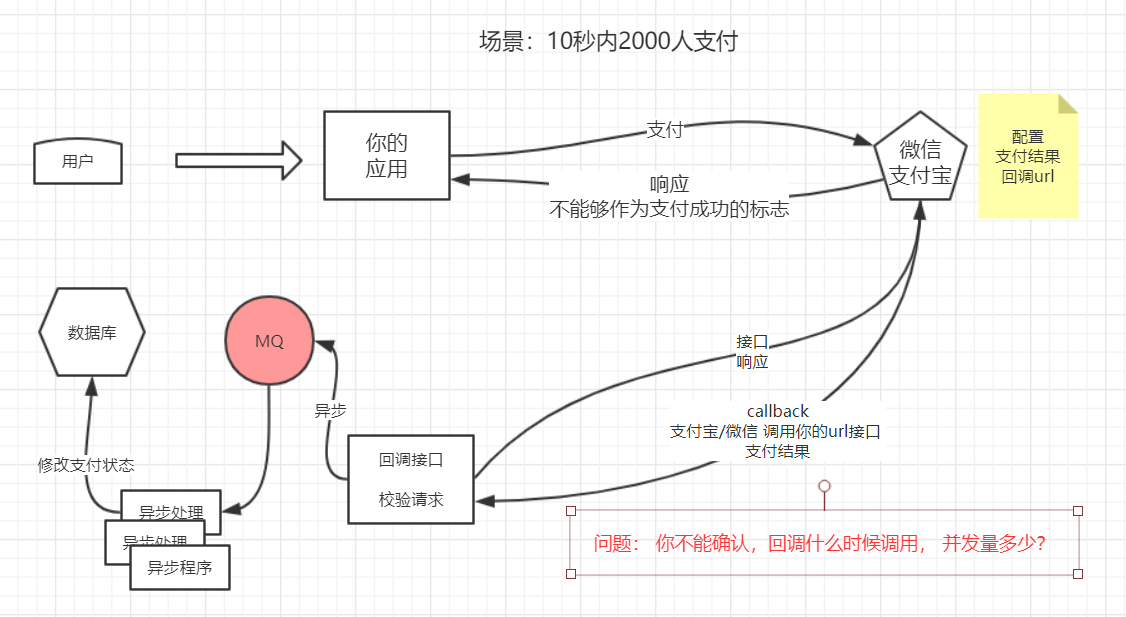
**系统解耦**： 调用者不关心执行结果 – 异步处理

Topic主题 – 多个系统订阅 – 适合一个消息，需要有多个系统去处理，或者多个实例（组件功能）

SpringCloud 事件通知 、配置中心 spring cloud config  
拓展：配置中心—disconf、qconf、diemond、阿波罗…  


Queue队列 --- 平常用来解耦。消费者集群不会重复消费。（业务处理）

-- 支付宝/微信 支付接口对接



分布式事务 --- 可靠消息

1. 可靠消息发布 -- confirm确认机制

--- 消息持久化（大部分MQ都支持，但不是默认开启）

--- 生产者要支持重发（网络中断，没收到回执）注：不可避免的出现重复消息

--- MQ的消息回执，发布确认（大部分都支持）

2. 可靠消息消费 --- ack机制

--- MQ要支持ack机制。mq分发消息到客户端，是否自动删除MQ中的这条消息

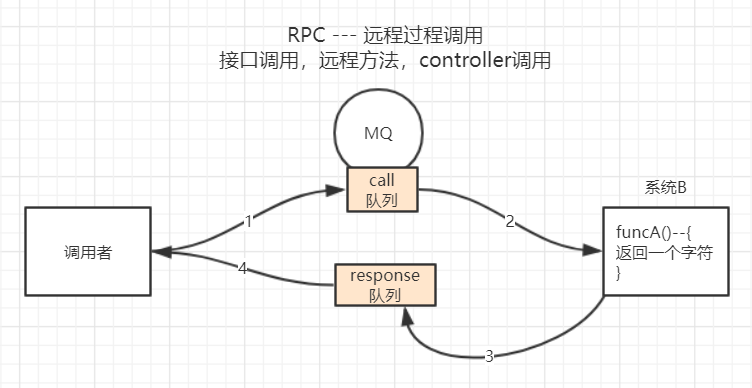
--- 消费者要开启手动ACK机制（正确且完成业务逻辑）。

--- ack 一般有三种套路：1. 删除MQ消息 2. MQ重发（重发次数限制，rabbitmq不支持次数限制，需要你自己去控制） 3. 死信队列

--- 幂等性的保证。重复消息不能够重复处理(记录处理过的消息)

场景： **MQ-RPC**RPC： 计算机/进程 之间 进行数据交互的手段

http、dubbo。接口rmi --- 远程过程调用Remote Procedure Call



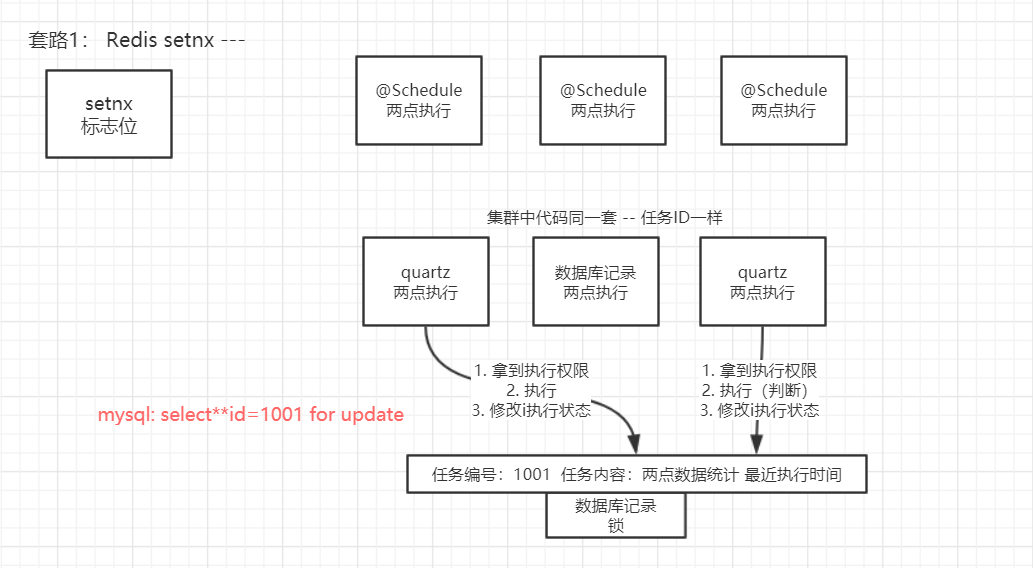
延时消息 - **MQ 任务调度** --- (一段代码/方法 多久触发一次)

--- 任务执行—调度

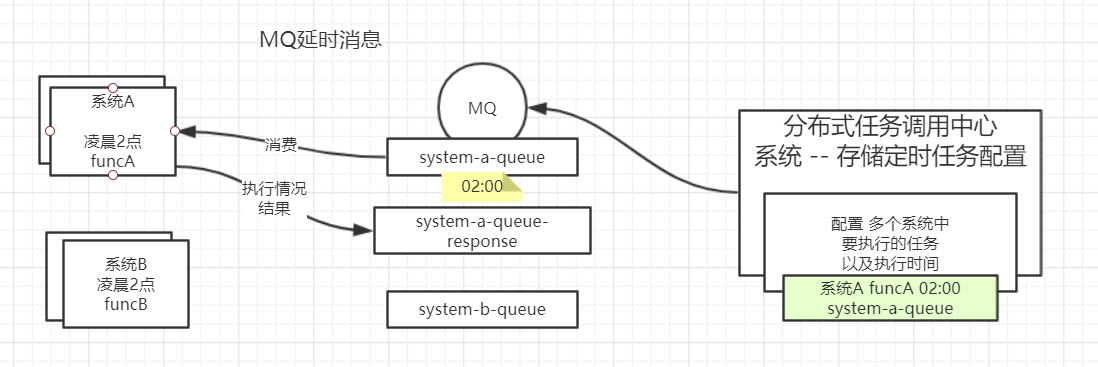
* 线程池、Timer
* @ Scheduled --- spring提供

---- 单机 ---单点问题：每天定时一次 --- 订单报表、每天用户数据统计等等

--- 集群 --- 最主要 重复执行 。而非并发执行

* Quartz集群，可以重复执行 – 利用数据库去记录执行的情况
* 分布式任务调度中心（基于MQ实现）

<http://www.xuxueli.com/xxl-job>



**延时消息：**

Activemq rocketmq 自带

Rabbitmq 需要自己折中实现 ---- 结合消息过期和死信队列去实现

Kafka 没考虑这些设计

