1、面试题

分布式服务接口的幂等性如何设计（比如不能重复扣款）？

2、面试官心里分析

从这个问题开始，面试官就已经进入了实际的生产问题的面试了

一个分布式系统中的某个接口，要保证幂等性，该如何保证？这个事儿其实是你做分布式系统的时候必须要考虑的一个生产环境的技术问题。啥意思呢？

你看，假如你有个服务提供一个接口，结果这服务部署在了5台机器上，接着有个接口就是付款接口。然后人家用户在前端上操作的时候，不知道为啥，总之就是一个订单不小心发起了两次支付请求，然后这俩请求分散在了这个服务部署的不同的机器上，好了，结果一个订单扣款扣两次？尴尬了。。。

或者是订单系统调用支付系统进行支付，结果不小心因为网络超时了，然后订单系统走了前面我们看到的那个重试机制，咔嚓给你重试了一把，好，支付系统收到一个支付请求两次，而且因为负载均衡算法落在了不同的机器上，尴尬了。。。

所以你肯定得知道这事儿，否则你做出来的分布式系统恐怕容易埋坑

网络问题很常见，100次请求，都ok；1万次，可能1次是超时会重试；10万，10次；100万，100次；如果有100个请求重复了，你没处理，导致订单扣款2次，100个订单都扣错了；每天被100个用户投诉；一个月被3000个用户投诉

我们之前生产就遇到过，是往数据库里写入数据，重复的请求，就导致我们的数据经常会错，出现一些重复数据，就会导致一些问题

3、面试题剖析

这个不是技术问题，这个没有通用的一个方法，这个是结合业务来看应该如何保证幂等性的，你的经验。

所谓幂等性，就是说一个接口，多次发起同一个请求，你这个接口得保证结果是准确的，比如不能多扣款，不能多插入一条数据，不能将统计值多加了1。这就是幂等性，不给大家来学术性词语了。

其实保证幂等性主要是三点：

（1）对于每个请求必须有一个唯一的标识，举个例子：订单支付请求，肯定得包含订单id，一个订单id最多支付一次，对吧

（2）每次处理完请求之后，必须有一个记录标识这个请求处理过了，比如说常见的方案是在mysql中记录个状态啥的，比如支付之前记录一条这个订单的支付流水，而且支付流水采

（3）每次接收请求需要进行判断之前是否处理过的逻辑处理，比如说，如果有一个订单已经支付了，就已经有了一条支付流水，那么如果重复发送这个请求，则此时先插入支付流水，orderId已经存在了，唯一键约束生效，报错插入不进去的。然后你就不用再扣款了。

（4）上面只是给大家举个例子，实际运作过程中，你要结合自己的业务来，比如说用redis用orderId作为唯一键。只有成功插入这个支付流水，才可以执行实际的支付扣款。

要求是支付一个订单，必须插入一条支付流水，order\_id建一个唯一键，unique key

所以你在支付一个订单之前，先插入一条支付流水，order\_id就已经进去了

你就可以写一个标识到redis里面去，set order\_id payed，下一次重复请求过来了，先查redis的order\_id对应的value，如果是payed就说明已经支付过了，你就别重复支付了

然后呢，你再重复支付这个订单的时候，你写尝试插入一条支付流水，数据库给你报错了，说unique key冲突了，整个事务回滚就可以了

来保存一个是否处理过的标识也可以，服务的不同实例可以一起操作redis。