每个工程师都应该了解的：聊聊幂等

2017-11-24 朱赟

每个工程师都应该了解的：聊聊幂等
朱赟
- 00:00 / 06:42

什么是幂等（Idempotency）呢？简单来说，一个操作如果多次任意执行所产生的影响，均与一次执行的影响相同，我们就称其为幂等。

这样说来，似乎很容易理解；但要知道这样的定义，其实是一个语义范畴对行为结果的定义。

如何用语法和规则去确保行为能达到这个结果，往往需要很谨慎地设计和实现。实际系统中，幂等是一个极为重要的概念，无论是在大型互联网应用还是企业级架构中，都能见到 REST API 被越来越多地采用，而正确实现幂等，往往是 API 中最难的技术点之一。

先说说为什么重要，我来举一个简单易懂的例子。

比如，你要处理一次电商网站收款或者付款的交易。当你给微信支付发送这个付款请求后，一个顺利的场景是不会有任何错误发生的，微信支付收到你的付款请求，处理所有转账，然后返回一个 HTTP 200 消息表示交易完成。

那如果发出请求后，有个请求超时，你再也没有收到关于这个请求是成功还是失败的回执，又该如何呢？

这里就有很多种可能的情况。

1. 这个请求在到达微信支付端前就已经发生超时，微信支付从来没有收到这样的请求。
2. 这个请求到达微信支付端，但是支付交易失败，这时发生超时，微信支付收到这样的请求，但没有处理成功。
3. 这个请求到达微信支付端，并且支付交易成功，这时发生超时，微信支付收到这样的请求，处理成功，但是没有回执。
4. 这个请求到达微信支付端，并且支付交易成功，并且发回回执，然而因为网络原因回执丢失，客户端超时，微信支付收到这样的请求，处理成功，发出回执，但是客户没有收到。

人们很直观的想法，也是现实中开发者最常见的做法就是：重新提交一次支付请求。但是这样做有一个潜在的问题：请求超时是上面的哪一种情况，会不会引发多次支付的可能性？

这就涉及系统中的幂等是如何实现的了。

那么幂等又该如何实现呢？

首先来看一下幂等的定义：多次执行所产生的影响均与一次执行的影响相同。简言之，你需要一个去重的机制。这往往有很多不同的实现方法，但是有两个很关键的因素。

第一个因素是幂等令牌（Idempotency Key）。客户端和服务器端通过什么方式来识别，这实际上是同一个请求或是同一个请求的多次尝试。这往往需要双方有一个既定的协议，比如账单号或者交易令牌，这种在同一个请求上具备唯一标识的元素，这种元素通常由客户端生成。

第二个因素是确保唯一性（Uniqueness Guarantee）。服务器端用什么机制去确保同一个请求一定不会被处理两次，也就是微信支付如何确保，同一笔交易不会因为客户端发送两次请求就被处理多次。

最常用的做法是利用数据库。比如把幂等令牌所在的数据库表的列作为唯一性索引。这样，当你试图存储两个含有同样令牌的请求时，必定有一个会报错。注意，简单的读检查并不一定成功，因为读与读之间会有竞争条件（Race Condition），因此还是有可能出错。

一个系统能正确处理和实现上面的两个要素，基本就达到了幂等的需求。那么，现实系统中常见的问题都出在哪里呢？

一是幂等令牌什么时候产生，怎样产生。这一点很重要。拿上面的例子来说，就算微信支付可以保证，每一个请求对应的支付交易一定只会被处理一次，但是这个请求的多次重复，一定要共有微信可以识别的某个标识。

假如客户端对同一笔交易多次请求，产生的幂等令牌并不相同，那么无论你其余的地方多么完美，都不可能保证“一个操作如果具有任意多次执行，所产生的影响均与一次执行的影响相同”。

二是令牌有没有被误删的可能。这是上面问题的一个特殊情况。幂等令牌是由客户端生成的，那如果生成的令牌在被使用后（一次微信支付请求中使用了），不小心因为数据库回滚（DB Rollback）等原因被删除了，那么客户端就不知道自己其实已经发过一次请求。这就有可能生成一个新的账单，并产生全新的令牌，而服务端对此则一无所知。

三是各种竞争条件。我在前面讲过，用数据库的读检查来确保唯一性经常因为竞争而不生效，其实一个需要幂等的系统中，保证唯一性的各个环节和实现，都要考虑竞争条件（Race Condition）。

四是对请求重试的处理。这大部分是服务器端要做的工作。一个常见的方法是：区分正在处理的请求、处理成功和处理失败的请求。这样当客户端重试的时候，根据情况或者直接返回，或者再次处理。这就好像之前提到的微信支付的例子，微信支付服务需要知道每一笔交易的处理情况，只有这样，当面对再次转账请求时，才能知道应该用什么方式去处理相应的请求。

五是一个系统中需要多层幂等。这是什么意思呢？A 发送请求给 B，B 处理的一部分是要发送请求给另一个系统 C，C 在处理的过程中还可能需要发请求给另一个系统 D ..... D 处理完了返回给 C，C 返回给 B，B 返回给 A。在这个链条中，如果 A、B、C、D 中任何一个系统没有正确实现幂等，也就是出现了“幂等漏洞”，那么一次请求还是有可能被多次执行，产生区别于一次执行的影响。

今天我和你讨论了架构设计中的幂等概念。我们聊了什么是幂等，幂等的应用场景，如何实现一个幂等功能，以及幂等系统中容易出现的问题。

现在回到文章的开头，什么是幂等？一个操作如果任意多次执行所产生的影响均与一次执行的影响相同，我们就称为幂等。这是一个语义范畴上对行为结果的定义，只有当你把实现中所有的细节都做对了，你才能得到想要的效果。任何一个地方设计有漏洞，或是实现上有 Bug，系统都会出现这样或那样的问题。

J

2017-11-24

如何测试是否达到幂等呢？是否存在一些方法论呢？特别是在大量使用开源软件等第三方技术和平台的时候，如果不是很清楚里面的坑，心里特别没底，但是我们没有那么多精力深入每一个细节。

幻想

2017-11-24

幂等操作确实很重要，除了文章支付的例子之外，像购物车占用库存操作，也是需要幂等的，不然可能出现超卖少卖现象。

huangzhimim

2017-11-30

第一次听到这样的概念，学到了

alexyugn

2017-12-04

做消息队列的consumer时候，特别要重视幂等信，保证相同的消息不论监听到多少次，也只能做处理一次

myaniu

2017-11-28

6年前曾设计了一个发短信然后银联自动给校园一卡通充值的系统，当时就是使用消息驱动，基于事物ID的幂等性，超时重发机制处理错误，最终效果不错。

小沫

2017-11-27

重复执行工单，多次下发操作数据 也会出现幂等情况。需要增加操作令牌以保证同一个资源数据只能操作一次

wangtie

2017-11-24

学习了，分享给小伙伴们看看

马洪博

2018-06-10

这里说的数据库“竞争条件”就是“脏读”吧。打个比方：
小张和小华同时喜欢上小芳，小张在探得小芳未婚后首先展开追求，并确定了关系。小华在几经打探确定小芳未婚后也打算采取行动，但在他准备行动的过程中小张和小芳闪婚了，使得上次的打探结果无效。
简单来说大概是“你读的时候并不知道别人也已经读过并已经采取了一些行动，只是行动尚未达成，对你不可见而已”

王岩

2017-11-27

不光是互联网支付，其实企业内部的对外支付也是一样，只要把微信服务端替换成银行就行了

bluze

2017-11-24

其实 我是第一次听说幂等的概念，虽然平常遇到很多幂等的问题

王宇熙

2018-05-10

例子中，幂等令牌的生成，不应该客户端完成，客户端只能保证单机幂等令牌的唯一性，不能保证全站幂等令牌的唯一性。

周丽洁

2018-05-04

问一个细节问题，为什么不能读检查而要靠写入报错来判断唯一性？因为我认为按一般的逻辑，是去数据库里面查有没有这个唯一值。不太明白读与读之间有竞争这个原因。谢谢回复。

gggjtg

2018-03-13

能不能举个例子解释一下什么是竞争条件

walt

2017-12-24

大并发测试幂等