编程中的幂等性 —— HTTP幂等性

计算机网络 yan 2015 年 7 月 21 日 8,652 ℃ 0评论 已收录

> 幂等（idempotent、idempotence）是一个数学与计算机学概念，常见于抽象代数中。
>
> 在编程中.**一个幂等操作的特点是其任意多次执行所产生的影响均与一次执行的影响相同**。幂等函数，或幂等方法，是指可以使用相同参数重复执行，并能获得相同结果的函数。这些函数不会影响系统状态，也不用担心重复执行会对系统造成改变。例如，“getUsername()和setTrue()”函数就是一个幂等函数.更复杂的操作幂等保证是利用唯一交易号(流水号)实现.
>
> ——百度百科

什么是幂等性(Idempotence)？

> Methods can also have the property of “idempotence” in that (aside from error or expiration issues) the side-effects of N > 0 identical requests is the same as for a single request.
>
> ——HTTP/1.1规范中幂等性的定义

从定义上看，HTTP方法的幂等性是指一次和多次请求某一个资源应该具有同样的副作用。说白了就是，**同一个请求，发送一次和发送N次效果是一样的！**幂等性是分布式系统设计中十分重要的概念，而HTTP的分布式本质也决定了它在HTTP中具有重要地位。下面将以HTTP中的幂等性做例子加以介绍。

简单示例

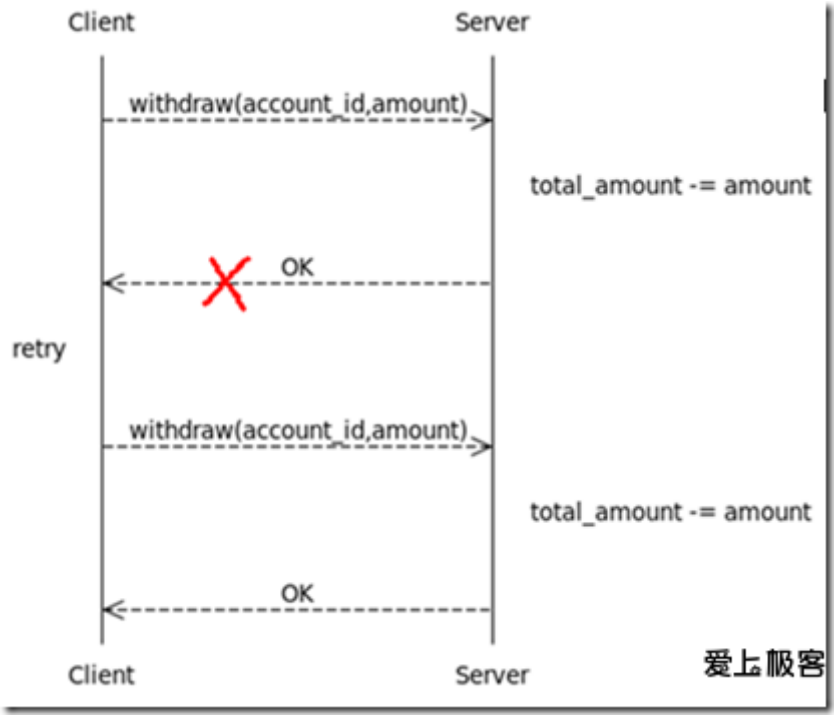
假设有一个从账户取钱的远程API（可以是HTTP的，也可以不是），我们暂时用类函数的方式记为：

```
1 bool withdraw(account_id, amount)
```

withdraw的语义是从account_id对应的账户中扣除amount数额的钱；如果扣除成功则返回true，账户余额减少amount；如果扣除失败则返回false，账户余额不变。

值得注意的是：和本地环境相比，**我们不能轻易假设分布式环境的可靠性。**

所以问题来了，一种典型的情况是withdraw请求已经被服务器端正确处理，但服务器端的返回结果由于网络等原因被掉丢了，导致客户端无法得知处理结果。如果是在网页上，一些不恰当的设计可能会使用户认为上一次操作失败了，然后刷新页面，这就导致了withdraw被调用两次，账户也被多扣了一次钱。如图所示：



解决方案一：采用分布式事务，通过引入支持分布式事务的中间件来保证withdraw功能的事务性。分布式事务的优点是对于调用者很简单，复杂性都交给了中间件来管理。缺点则是一方面架构太重量级，容易被绑在特定的中间件上，不利于异构系统的集成；另一方面分布式事务虽然能保证事务的ACID性质，而但却无法提供性能和可用性的保证。

解决方案二：幂等设计。我们可以通过一些技巧把withdraw变成幂等的，比如：

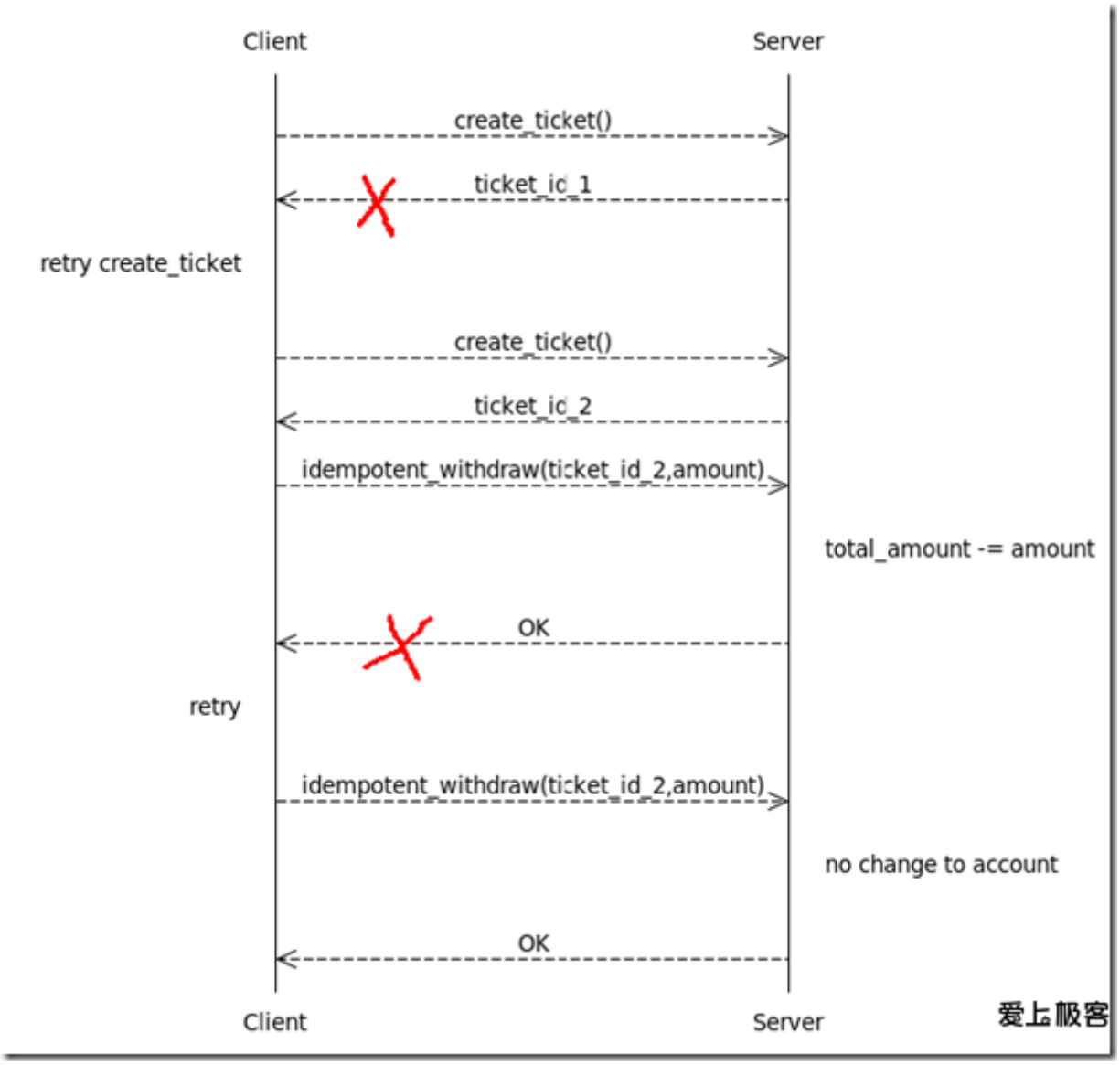
```
int create_ticket()
bool idempotent_withdraw(ticket_id, account_id, amount)

```

create_ticket的语义是**获取一个服务器端生成的唯一的处理号ticket_id，它将用于标识后续的操作。**idempotent_withdraw和withdraw的区别在于关联了一个ticket_id，一个ticket_id表示的操作至多只会被处理一次，每次调用都将返回第一次调用时的处理结果。这样，idempotent_withdraw就符合幂等性了，客户端就可以放心地多次调用。

基于幂等性的解决方案中一个完整的取钱流程被分解成了两个步骤：1.调用create_ticket()获取ticket_id；2.调用idempotent_withdraw(ticket_id, account_id, amount)。虽然create_ticket不是幂等

的，但在这种设计下，它对系统状态的影响可以忽略，加上idempotent_withdraw是幂等的，所以任何一步由于网络等原因失败或超时，客户端都可以重试，直到获得结果。如图所示：



和分布式事务相比，幂等设计的优势在于它的轻量级，容易适应异构环境，以及性能和可用性方面。在某些性能要求比较高的应用，幂等设计往往是唯一的选择。

HTTP的幂等性

本文主要以HTTP GET、DELETE、PUT、POST四种方法为主进行语义和幂等性的介绍。

HTTP GET方法用于获取资源，不应有副作用，所以是幂等的。比如：GET http://www.bank.com/account/123456，不会改变资源的状态，不论调用一次还是N次都没有副作用。请注意，这里强调的是一次和N次具有相同的副作用，而不是每次GET的结果相同。GET http://www.news.com/latest-news这个HTTP请求可能会每次得到不同的结果，但它本身并没有产生任何副作用，因而是满足幂等性的。

HTTP DELETE方法用于删除资源，有副作用，但它应该满足幂等性。比如：DELETE http://www.forum.com/article/4231，调用一次和N次对系统产生的副作用是相同的，即删掉id为4231的帖

子；因此，调用者可以多次调用或刷新页面而不必担心引起错误。

HTTP POST方法用于创建资源，所对应的URI并非创建的资源本身，而是去执行创建动作的操作者，有副作用，不满足幂等性。比如：POST http://www.forum.com/articles的语义是在http://www.forum.com/articles下创建一篇帖子，HTTP响应中应包含帖子的创建状态以及帖子的URI。两次相同的POST请求会在服务器端创建两份资源，它们具有不同的URI；所以，POST方法不具备幂等性。

HTTP PUT方法用于创建或更新操作，所对应的URI是要创建或更新的资源本身，有副作用，它应该满足幂等性。比如：PUT http://www.forum/articles/4231的语义是创建或更新ID为4231的帖子。对同一URI进行多次PUT的副作用和一次PUT是相同的；因此，PUT方法具有幂等性。

对前文示例进行改进

利用Web API的形式实现前面所提到的取款功能。

1、用POST /tickets来实现create_ticket；

2、用PUT /accounts/account_id/ticket_id&amount=xxx来实现idempotent_withdraw。

值得注意的是严格来讲amount参数不应该作为URI的一部分，真正的URI应该是/accounts/account_id/ticket_id，而amount应该放在请求的body中。这种模式可以应用于很多场合，比如：论坛网站中防止意外的重复发帖。

电商中遇到的问题

如何防范 POST 重复提交

HTTP POST 操作既不是安全的，也不是幂等的（至少在HTTP规范里没有保证）。当我们因为反复刷新浏览器导致多次提交表单，多次发出同样的POST请求，导致远端服务器重复创建出了资源。

所以，对于电商应用来说，第一对应的后端 WebService 一定要做到**幂等性**，第二服务器端收到 POST 请求，在操作成功后**必须302跳转到另外一个页面**，这样即使用户刷新页面，也不会重复提交表单。

把分布式事务分解为具有幂等性的异步消息处理

电商的很多业务，考虑更多的是 BASE（即Basically Available、Soft state、和Eventually consistent），而不是 ACID（Atomicity、Consistency、Isolation和 Durability）。即为了满足高负载的用户访问，我们可以容忍短暂的数据不一致。那怎么做呢？

第一，不做分布式事务，代价太大。
第二，不一定需要实时一致性，只需要保证最终的一致性即可。
第三，“通过状态机和严格的有序操作，来最大限度地降低不一致性”。
第四，最终一致性（Eventually Consistent）通过异步事件做到。

如果消息具有操作幂等性，也就是一个消息被应用多次与应用一次产生的效果是一样的话，那么把不需要同步执行的事务交给异步消息推送和订阅者集群来处理即可。假如消息处理失败，那么就消息重播，由于幂等性，应用多次也能产生正确的结果。

实际情况下，消息很难具有幂等性，解决方法是使用另一个表记录已经被成功应用的消息，即消息队列和消息应用状态表一起来解决问题。

总结

上面简单介绍了幂等性的概念，用幂等设计取代分布式事务的方法，以及HTTP主要方法的语义和幂等性特征。其实，如果要追根溯源，幂等性是数学中的一个概念，表达的是N次变换与1次变换的结果相同，有兴趣的读者可以从Wikipedia上进一步了解。

3

分布式 幂等性

« 随机函数扩大，如rand5()构造rand7()

分清JavaEE，Tomcat，JVM，JRE，JDK，SDK基本概念




Java Web分布式集群搭建（三）——Session同 | Java Web分布式集群搭建（二）—— | Java Web分布式集群搭建（一）——Mysql集 | 《大型网站技术架构》读书笔记

- Java Web分布式集群搭建（三）——Session同步
- Java Web分布式集群搭建（二）——Apache+Tomcat集群负载均衡
- Java Web分布式集群搭建（一）——Mysql集群
- 《大型网站技术架构》读书笔记
- 网格计算、分布式计算、集群、云计算介绍和比较
- WebSocket介绍，与Socket的区别
- 正向代理、反向代理、透明代理以及CDN的区别
- 数据链路层的点对点通信（PPP）和广播通信（Ethernet）的研究

发表我的评论

写点什么...

表情 有人回复时邮件通知我 提交评论