腾讯电商部门二面：如何保证幂等性?

你好，我是猿java。

在日常开发中，我们经常听到xxx要保证幂等性，在这篇文章中，我们将分析一道腾讯电商部门的二面题：如何保证幂等性？通过本文，我们将揭开幂等性概念的神秘面纱，以及在系统中实施幂等性的最佳做法。

什么是幂等性?

在数学中，若对于某个运算 总存在`f(f(x))=f(x)`，则称该运算为幂等操作，例如，绝对值函数`f(x)=|x|`就是幂等的，下面为两个幂等性的简单数学示例：

```
x + 0;
x = 6;
```

在第一个示例中，无论执行多少次，加零操作都不会改变结果，在第二个示例中，无论执行该操作多少次，x始终为6，这两个示例都描述了幂等操作。

在计算机科学中，幂等性（Idempotence）主要用于描述某些操作或函数在多次执行后的效果，具体来说，幂等操作在重复执行多次后，其结果与执行一次的结果相同。这一原则在分布式系统和 API中尤为关键，有助于在网络问题、请求重试或重复请求等情况下保持一致性和可预测性。

为什么需要幂等性?

幂等性在 API中很重要，因为如果网络中断，资源可能会被多次调用，在这种情况下，非幂等操作可能会创建额外的资源或意外更改它们，从而导致重大的意外副作用，如果要求数据的准确性时，非幂等性会带来重大风险。

我们可以想象一下：用户发送请求，将 1000元从账户A转移到账户B，由于网络延迟或其他因素导致该请求被发送两次，如果没有幂等性，API将处理这两个请求，从而导致 2000元被转出，这种转账体验肯定是很糟糕的。

因此，幂等性对于确保以下三点至关重要：

* 一致性：即使面临请求重复或重试，系统也能保持可预测的状态。
* 错误处理：幂等操作简化了错误处理，因为客户端可以安全地重试请求，而不会产生意外的副作用。
* 容错：幂等 API 可以更好地应对网络问题和其他中断，确保更可靠的用户体验。

幂等与安全
=====

“幂等方法”和“安全方法”这两个概念经常被混淆，安全方法只会读取，从不写入，因此它不会更改返回的值，回到我们之前的例子：

```
x + 0;
x = 6;
```

* 第一个方法，加零操作，每次都返回相同的值，所以它是幂等的，加零操作对该值本身没有影响，因此它也是安全的；
* 第二个示例每次都会返回相同的值，因此它是幂等的，但并不安全，因为如果x在操作运行之前不是 6，它将更改 x的值；

因此，所有安全方法都是幂等的，但并非所有幂等方法都是安全的。

REST中的幂等方法
==========

在 REST API中，常常使用GET、HEAD、PUT、DELETE、OPTIONS 和 TRACE 等方法进行交互，那么它们中间哪些是幂等的？哪些是非幂等的？下面我们将一一介绍。

POST 方法

POST用于将数据提交到指定资源，通常用于在服务器上创建或更新内容，POST 本身并不具有幂等性，这意味着多次发送相同的 POST 请求可能会导致不同的结果或创建提交数据的多个实例。

因此，开发人员在处理 POST请求、实施幂等性密钥或其他保护措施等机制时需要谨慎，以防止在提交或重试重复请求的情况下产生意外的副作用。

GET方法

GET方法是用于从 Web上获取特定资源，当客户端向服务器发送 GET请求时，它要求服务器检索并返回与指定资源（如网页、图像或文档）关联的信息。GET请求被设计为只读，因此它具备幂等性，这意味着对同一资源发出多个相同的 GET请求将产生相同的结果，而不会引起任何额外的更改。

HEAD方法

HEAD用于检索有关特定资源的元数据，而无需实际下载资源的内容，与 GET方法类似，HEAD方法用于请求有关资源的信息，但它仅返回 HTTP标头，其中包括内容类型、内容长度和上次修改日期等信息。当客户端想要检查资源是否存在或检索其元数据而不产生下载整个资源的成本时，此方法特别有用。

HEAD 方法是幂等的，这意味着在同一资源上使用 HEAD的多个请求将产生相同的结果，而不会产生任何副作用。

PUT方法

PUT用于使用客户端提供的新数据更新或替换服务器上的现有资源。该请求包含一个唯一标识符，例如 URI，服务器使用该标识符来查找目标资源。客户端还提供一个有效负载，其中包含要应用于资源的更新数据。

PUT 本质上是幂等的，这意味着发出多个相同的 PUT请求将与发出单个请求具有相同的效果。如果指定的 URI中已存在资源，则服务器将用新数据替换它。

如果资源不存在，服务器可能会创建它，具体取决于实现。

PATCH方法

PATCH用于部分更新服务器上的资源，与 PUT不同，PATCH仅更改某些资源，它的幂等性取决于操作的实现和语义。

虽然如果以一致和确定性的方式应用更改，则 PATCH 可以是幂等的，但它本质上并不像 GET、PUT 和 DELETE 那样具有幂等性。在实践中，通过仔细设计 API和底层操作，可以使 PATCH具有幂等性，确保重复的 PATCH请求产生与单个请求相同的结果。

DELETE方法

DELETE用于从服务器中删除指定的资源，DELETE方法被认为是幂等的，因为多次执行同一请求会导致相同的结果。在初始成功删除后，对同一 URI的任何后续 DELETE请求都不应对服务器的状态产生额外影响，因为资源不再存在。

TRACE方法

TRACE允许客户端检索服务器上特定资源的请求和响应消息的诊断表示形式，当客户端发送 TRACE请求时，服务器会返回整个 HTTP请求（包括标头）作为响应正文。此方法主要用于调试目的，使开发人员能够检查和诊断请求和响应周期中的潜在问题。

TRACE 被认为是幂等的，因为根据设计，它不会更改服务器上的资源或其状态。当发出多个 TRACE 请求时，服务器将始终如一地返回相同的请求数据，使其成为一个安全且可预测的操作。但是，出于安全考虑，TRACE通常会在 Web服务器上禁用，以防止潜在的信息泄露。

从上述介绍可以看出：REST API中使用的大多数方法都是幂等的。例外情况是 POST。但是，值得记住的是，如果使用不当，GET 和 HEAD 等其他方法仍然可以以非幂等方式调用。开发人员可以通过遵循 REST原则来避免这种情况。

幂等性的实现方案

========

实现幂等性的方法因具体应用场景而异，以下是一些常见的幂等性实现方案：

使用唯一标识符

使用唯一标识符，这种方法常用于API设计中，尤其是金融交易和订单系统中。关于唯一标识（Idempotency Key）的生成可以参考：[新来的技术女总监，推荐的分布式 ID真香！](http://cxyroad.com/”https://mp.weixin.qq.com/s?__biz=MzIwNDAyOTI2Nw==&mid=2247495804&idx=1&sn=3ec8b6ea302bf19a1fe4aeb7f7273248&chksm=96c4dc40a1b35556b2064d8adfdba523cd15c6f2f98e8e7f4dc855c97fd3dba8db6010040939&token=802037199&lang=zh_CN#rd”) 这篇文章。

如何使用唯一标识符来保证幂等？下面给出了一个通用的步骤：

* 客户端在发出请求时生成一个唯一标识符（Idempotency Key）；
* 服务器在处理请求时检查这个唯一标识符是否已经处理过；
* 如果已经处理过，则返回之前的响应，否则，处理请求并存储响应结果与唯一标识符；

优点

* 使用唯一标识符，确保每个请求只被处理一次，即使客户端因为网络原因重复发送请求。

数据库约束

数据库的唯一约束或主键，也是日常开发中用来确保操作幂等性的一种手段。下面也给出了其使用步骤：

* 对于需要幂等的操作，如插入数据，确保插入的数据具有唯一的标识符（如主键）。
* 数据库会自动保证相同的插入操作只执行一次。

优点

* 数据库约束，利用数据库的特性保证幂等性，因此实现比较简单。

乐观锁

在更新操作中，通常使用乐观锁来确保只有在特定条件下才执行更新。使用乐观锁保证幂等的步骤为：

* 每次读取数据时，读取其版本号。
* 在更新数据时，检查版本号是否与之前读取的一致。
* 如果一致，则执行更新并增加版本号，否则，更新失败，提示重试。

优点

* 使用乐观锁，可以防止并发更新导致的数据不一致问题。

幂等操作设计

通过设计操作本身的逻辑，使其具有幂等性，其步骤为：

* 确保操作本身是幂等的，例如，将数据更新为特定值而不是增加或减少值。
* 对于删除操作，可以先检查资源是否存在，再执行删除操作。

优点
–通过人为设计逻辑确保操作幂等性，简化了实现过程。

缓存机制

使用缓存机制存储请求结果，以防止重复处理，其步骤为：

* 在处理请求之前，检查缓存中是否已有结果。

* 如果有，直接返回缓存结果；否则，处理请求并将结果存入缓存。

优点

* 减少重复计算，提升系统性能。

幂等性在实践中的案例

幂等性在实际应用中有很多成功的案例，以下是几个典型的应用场景：

电商系统中的订单处理

在电商系统中，订单处理通常包括创建订单、支付订单、更新订单状态等多个步骤。为了确保系统的一致性和可靠性，这些操作通常设计为幂等的。例如，在支付订单时，每个支付请求都带有唯一的交易ID，支付服务通过检查交易ID来确保每个订单只被支付一次。

分布式数据库的事务处理

在分布式数据库中，事务处理通常需要跨多个节点进行协调。幂等性可以确保即使同一个事务被多次提交，数据库的最终状态也是一致的。例如，在使用分布式事务协议（如二阶段提交）时，幂等性可以确保事务的提交和回滚操作在多次执行时不会引起数据的不一致。

微服务架构中的服务调用

在微服务架构中，服务之间的调用通常通过网络进行，网络的不可靠性可能导致请求失败或超时。通过设计服务的API为幂等的，可以确保服务的调用在重试时不会引起副作用。例如，在用户服务中更新用户信息时，通过使用PUT方法并检查用户ID，确保更新操作是幂等的。

总结

本文分析了什么是幂等性以及保证幂等的一些常用方法，对于幂等性，一个核心的判断方式：操作多次和操作一次的结果是不是一样，如果一样就幂等，不一样就不幂等。

在日常开发中，我们一定要注意业务场景是否需要保证幂等，以免增加不必要的副作用。

学习交流
====

如果你觉得文章有帮助，请帮忙点个赞呗，或者公众号：猿java，持续输出硬核文章。

原文链接: https://juejin.cn/post/7388444606458019859