T与POST的对比解析

有了对HTTP的基本认识，我们再来看下Get和Post到底有何不同。

Get和Post是什么?

GET和POST是HTTP中的两个最基本的请求方法。在HTTP协议的早期版本中，GET是最初定义的方法，主要用于请求文档；然后POST方法被引入，以支持更复杂的交互模式。在1996年的HTTP/1.0（RFC 1945）中首次明确定义了GET和POST方法。

我们先来看一个标准的说法：

* GET 主要用于请求访问已经被URI（统一资源标识符，URL是它的一个子集）识别的资源。它的设计目的是从服务器上获取数据，适用于信息的检索。
* POST 主要用于向指定的资源提交数据，请求服务器进行处理（例如，提交表单或上传文件）。数据被包含在请求体中。它的设计目的是允许用户向服务器传输数据，适用于更新或创建资源。

我们再用一个比喻来加深理解：

使用GET请求就像是去学校图书馆借一本书，你告诉图书管理员你需要的书名（通过URL的查询字符串传递参数），图书管理员找到书后将其递给你。

使用POST请求则像是你写了一张点餐单（扫码点餐流行之前一般手写），然后把它放进了一个信封交给服务员，服务员就去协调厨房制作菜品了，当然服务员还会把做好的菜给你送过来。

基础区别

我们可以从使用形式上对他俩进行一个基础的对比。

项目	**Get请求**	**Post请求**
****参数位置**	请求的参数附加在URL之后	请求的参数放在请求体内
数据大小	URL长度存在限制，传递的数据量相对较小	理论上不受限，可以传递更多的数据。
数据缓存	可以被缓存，历史记录可以保存	不会被缓存，也不会留在浏览器历史记录中
后退/刷新	无害	数据会被重新提交
适用场景	适合公开的、无副作用的数据检索操作。	适合创建或更新资源。

HTTP作为一种文本传输协议，Get和Post的描述方式其实没有太大的差别。我这里还找了两张图，大家可以看看HTTP实际上是怎么完成Get和Post请求的。

首先看Get方法：

![](https://p3-juejin.byt 作类型operation，然后再根据类型解析data中的数据，最后按照协议中的操作类型进行相应的处理。

其实不仅很多的私有业务协议会完全基于HTTP Post进行构建，大名顶顶的gRPC也是基于HTTP的Post方法搞出来的。当然gRPC使用的HTTP和我们日常浏览网页使用的不太一样，gRPC使用的是HTTP/2，它具备长连接、多路复用、服务器推送、头部压缩等特性，这会带来更低的延迟、更少资源消耗和更高的吞吐量，可以满足gRPC的性能要求。同时gRPC的设计初衷是支持客户端和服务器之间的双向流式通信，而POST方法能够支持发送请求体（Body），这对于gRPC传输复杂的消息结构是必需的，再使用其它HTTP方法没有多少必要。

但是gRPC选择HTTP还有另一个考虑，大量的现有的网络基础设施（如代理、负载均衡器、API网关等）已经支持或可以较为容易地适应HTTP，即使是HTTP/2。gRPC基于HTTP可以更容易地融入现有IT环境，利用已有的服务治理工具和中间件进行监控、路由、安全控制等。

回到大量私有业务协议使用HTTP也是类似的，HTTP操作简单，网络兼容性好，可以方便的集成到现有的系统环境中，而且Post可以完全的覆盖各种业务操作需求。

另外HTTP协议规范也没有明确禁止 GET 请求中使用 HTTP Body 来传输数据。从技术角度讲，GET 请求可以携带 Body。但是很多客户端库、服务器软件

、中间件、浏览器以及开发工具可能并不支持或默认禁用 GET 请求中的Body，所以大家没有用Get来搞事情。

最后，我们做个简单的总结：

GET和POST被设计用来满足不同的网络交互需求，GET用于数据检索，主要目的是获取资源；Post用于数据提交，主要目的是创建和更新资源。

虽然POST请求看起来“更安全”，但无论是GET还是POST，本质上并非围绕安全性设计，要安全的传输，应该始终使用HTTPS，过加密传输层，保护数据不被中间人窃取或篡改。

Post具备高度的灵活性，它不仅适用于提交数据，也能用于资源的查询、删除等操作。这种灵活性使得POST成为许多私有业务协议和高性能通信协议（如gRPC）的首选基础。尽管Get请求理论上也能携带Body，但实际应用受到很多限制。

萤火架构，提升技术认知！

原文链接: https://juejin.cn/post/7351682240747503653