## 10-应该如何理解请求方法？

上一讲我介绍了HTTP的报文结构，它是由header+body构成，请求头里有请求方法和请求目标，响应头里有状态码和原因短语，今天要说的就是请求头里的请求方法。

### 标准请求方法

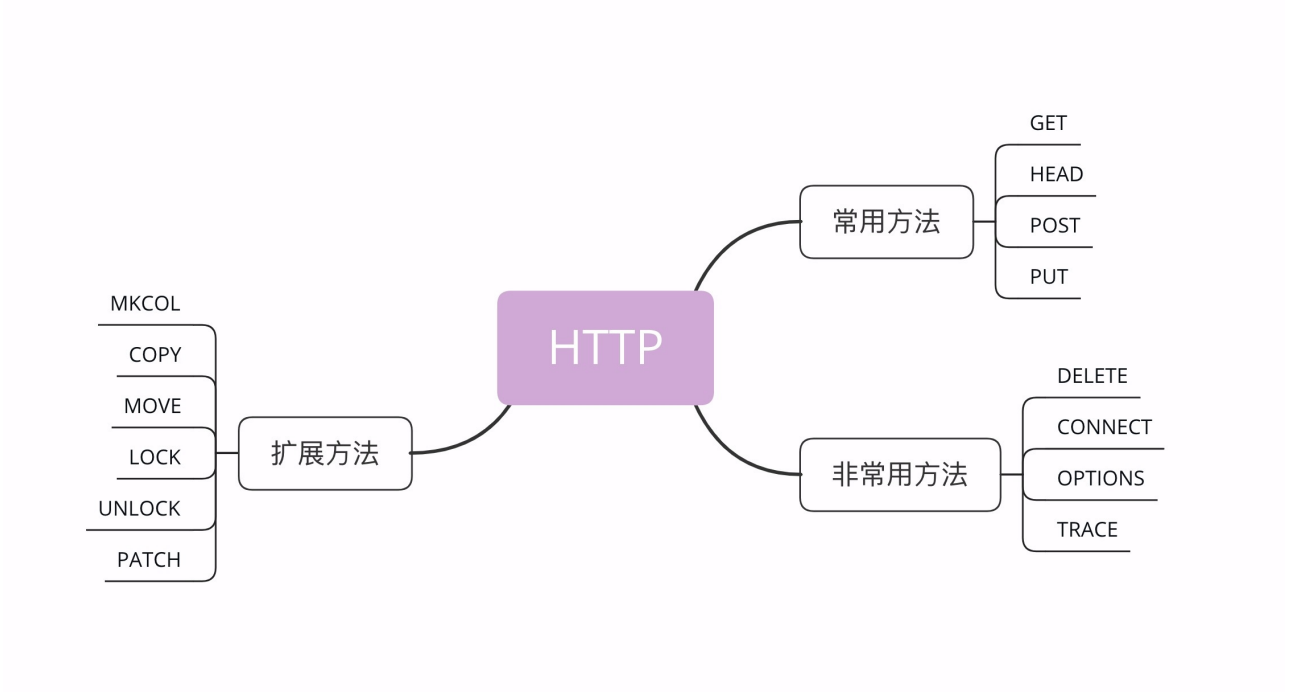
HTTP协议里为什么要有“请求方法”这个东西呢？

这就要从HTTP协议设计时的定位说起了。还记得吗？蒂姆·伯纳斯-李最初设想的是要用HTTP协议构建一个超链接文档系统，使用URI来定位这些文档，也就是资源。那么，该怎么在协议里操作这些资源呢？

很显然，需要有某种“动作的指示”，告诉操作这些资源的方式。所以，就这么出现了“请求方法”。它的实际含义就是客户端发出了一个“动作指令”，要求服务器端对URI定位的资源执行这个动作。

目前HTTP/1.1规定了八种方法，单词**都必须是大写的形式**，我先简单地列把它们列出来，后面再详细讲解。

1. GET：获取资源，可以理解为读取或者下载数据；
2. HEAD：获取资源的元信息；
3. POST：向资源提交数据，相当于写入或上传数据；
4. PUT：类似POST；
5. DELETE：删除资源；
6. CONNECT：建立特殊的连接隧道；
7. OPTIONS：列出可对资源实行的方法；
8. TRACE：追踪请求-响应的传输路径。




看看这些方法，是不是有点像对文件或数据库的“增删改查”操作，只不过这些动作操作的目标不是本地资源，而是远程服务器上的资源，所以只能由客户端“请求”或者“指示”服务器来完成。

既然请求方法是一个“指示”，那么客户端自然就没有决定权，服务器掌控着所有资源，也就有绝对的决策权力。它收到HTTP请求报文后，看到里面的请求方法，可以执行也可以拒绝，或者改变动作的含义，毕竟

HTTP是一个“协议”，两边都要“商量着来”。

比如，你发起了一个GET请求，想获取“/orders”这个文件，但这个文件保密级别比较高，不是谁都能看的，服务器就可以有如下的几种响应方式：

1. 假装这个文件不存在，直接返回一个404 Not found报文；
2. 稍微友好一点，明确告诉你有这个文件，但不允许访问，返回一个403 Forbidden；
3. 再宽松一些，返回405 Method Not Allowed，然后用Allow头告诉你可以用HEAD方法获取文件的元信息。

## GET/HEAD

虽然HTTP/1.1里规定了八种请求方法，但只有前四个是比较常用的，所以我们先来看一下这四个方法。

**GET**方法应该是HTTP协议里最知名的请求方法了，也应该是用的最多的，自0.9版出现并一直被保留至今，是名副其实的“元老”。

它的含义是请求**从服务器获取资源**，这个资源既可以是静态的文本、页面、图片、视频，也可以是由PHP、Java动态生成的页面或者其他格式的数据。

GET方法虽然基本动作比较简单，但搭配URI和其他头字段就能实现对资源更精细的操作。

例如，在URI后使用“#”，就可以在获取页面后直接定位到某个标签所在的位置；使用If-Modified-Since字段就变成了“有条件的请求”，仅当资源被修改时才会执行获取动作；使用Range字段就是“范围请求”，只获取资源的一部分数据。

**HEAD**方法与GET方法类似，也是请求从服务器获取资源，服务器的处理机制也是一样的，但服务器不会返回请求的实体数据，只会传回响应头，也就是资源的“元信息”。

HEAD方法可以看做是GET方法的一个“简化版”或者“轻量版”。因为它的响应头与GET完全相同，所以可以用在很多并不真正需要资源的场合，避免传输body数据的浪费。

比如，想要检查一个文件是否存在，只要发个HEAD请求就可以了，没有必要用GET把整个文件都取下来。再比如，要检查文件是否有最新版本，同样也应该用HEAD，服务器会在响应头里把文件的修改时间传回来。

你可以在实验环境里试一下这两个方法，运行Telnet，分别向URI“/10-1”发送GET和HEAD请求，观察一下响应头是否一致。

```
GET /10-1 HTTP/1.1
Host: www.chrono.com


HEAD /10-1 HTTP/1.1
Host: www.chrono.com
```

## POST/PUT

接下来要说的是**POST**和**PUT**方法，这两个方法也很像。

GET和HEAD方法是从服务器获取数据，而POST和PUT方法则是相反操作，向URI指定的资源提交数据，数据就放在报文的body里。

POST也是一个经常用到的请求方法，使用频率应该是仅次于GET，应用的场景也非常多，只要向服务器发送数据，用的大多数都是POST。

比如，你上论坛灌水，敲了一堆字后点击“发帖”按钮，浏览器就执行了一次POST请求，把你的文字放进报文的body里，然后拼好POST请求头，通过TCP协议发给服务器。

又比如，你上购物网站，看到了一件心仪的商品，点击“加入购物车”，这时也会有POST请求，浏览器会把商品ID发给服务器，服务器再把ID写入你的购物车相关的数据库记录。

PUT的作用与POST类似，也可以向服务器提交数据，但与POST存在微妙的不同，通常POST表示的是“新建”“create”的含义，而PUT则是“修改”“update”的含义。

在实际应用中，PUT用到的比较少。而且，因为它与POST的语义、功能太过近似，有的服务器甚至就直接禁止使用PUT方法，只用POST方法上传数据。

实验环境的“/10-2”模拟了POST和PUT方法的处理过程，你仍然可以用Telnet发送测试请求，看看运行的效果。注意，在发送请求时，头字段“Content-Length”一定要写对，是空行后body的长度：

```
POST /10-2 HTTP/1.1
Host: www.chrono.com
Content-Length: 17

POST DATA IS HERE

PUT /10-2 HTTP/1.1
Host: www.chrono.com
Content-Length: 16

PUT DATA IS HE
```

## 其他方法

讲完了GET/HEAD/POST/PUT，还剩下四个标准请求方法，它们属于比较“冷僻”的方法，应用的不是很多。

**DELETE**方法指示服务器删除资源，因为这个动作危险性太大，所以通常服务器不会执行真正的删除操作，而是对资源做一个删除标记。当然，更多的时候服务器就直接不处理DELETE请求。

**CONNECT**是一个比较特殊的方法，要求服务器为客户端和另一台远程服务器建立一条特殊的连接隧道，这时Web服务器在中间充当了代理的角色。

**OPTIONS**方法要求服务器列出可对资源实行的操作方法，在响应头的Allow字段里返回。它的功能很有限，用处也不大，有的服务器（例如Nginx）干脆就没有实现对它的支持。

**TRACE**方法多用于对HTTP链路的测试或诊断，可以显示出请求-响应的传输路径。它的本意是好的，但存在漏洞，会泄漏网站的信息，所以Web服务器通常也是禁止使用。

## 扩展方法

虽然HTTP/1.1里规定了八种请求方法，但它并没有限制我们只能用这八种方法，这也体现了HTTP协议良好的扩展性，我们可以任意添加请求动作，只要请求方和响应方都能理解就行。

例如著名的愚人节玩笑RFC2324，它定义了协议HTCPCP，即“超文本咖啡壶控制协议”，为HTTP协议增加了用来煮咖啡的BREW方法，要求添牛奶的WHEN方法。

此外，还有一些得到了实际应用的请求方法（WebDAV），例如MKCOL、COPY、MOVE、LOCK、UNLOCK、PATCH等。如果有合适的场景，你也可以把它们应用到自己的系统里，比如用LOCK方法锁定资源暂时不允许修改，或者使用PATCH方法给资源打个小补丁，部分更新数据。但因为这些方法是非标准的，所以需要为客户端和服务器编写额外的代码才能添加支持。

当然了，你也完全可以根据实际需求，自己发明新的方法，比如“PULL”拉取某些资源到本地，“PURGE”清理某个目录下的所有缓存数据。

## 安全与幂等

关于请求方法还有两个面试时有可能会问到、比较重要的概念：**安全**与**幂等**。

在HTTP协议里，所谓的“**安全**”是指请求方法不会“破坏”服务器上的资源，即不会对服务器上的资源造成实质的修改。

按照这个定义，只有GET和HEAD方法是“安全”的，因为它们是“只读”操作，只要服务器不故意曲解请求方法的处理方式，无论GET和HEAD操作多少次，服务器上的数据都是“安全的”。

而POST/PUT/DELETE操作会修改服务器上的资源，增加或删除数据，所以是“不安全”的。

所谓的“**幂等**”实际上是一个数学用语，被借用到了HTTP协议里，意思是多次执行相同的操作，结果也都是相同的，即多次“幂”后结果“相等”。

很显然，GET和HEAD既是安全的也是幂等的，DELETE可以多次删除同一个资源，效果都是“资源不存在”，所以也是幂等的。

POST和PUT的幂等性质就略费解一点。

按照RFC里的语义，POST是“新增或提交数据”，多次提交数据会创建多个资源，所以不是幂等的；而PUT是“替换或更新数据”，多次更新一个资源，资源还是会第一次更新的状态，所以是幂等的。

我对你的建议是，你可以对比一下SQL来加深理解：把POST理解成INSERT，把PUT理解成UPDATE，这样就很清楚了。多次INSERT会添加多条记录，而多次UPDATE只操作一条记录，而且效果相同。

## 小结

今天我们学习了HTTP报文里请求方法相关的知识，简单小结一下。

1. 请求方法是客户端发出的、要求服务器执行的、对资源的一种操作；
2. 请求方法是对服务器的“指示”，真正应如何处理由服务器来决定；
3. 最常用的请求方法是GET和POST，分别是获取数据和发送数据；
4. HEAD方法是轻量级的GET，用来获取资源的元信息；
5. PUT基本上是POST的同义词，多用于更新数据；
6. “安全”与“幂等”是描述请求方法的两个重要属性，具有理论指导意义，可以帮助我们设计系统。

## 课下作业

1. 你能把GET/POST等请求方法对应到数据库的“增删改查”操作吗？请求头应该如何设计呢？
2. 你觉得TRACE/OPTIONS/CONNECT方法能够用GET或POST间接实现吗？

欢迎你把自己的答案写在留言区，与我和其他同学一起讨论。如果你觉得有所收获，欢迎你把文章分享给你的朋友。

### 课外小贴士

01 Nginx 默认不支持 OPTIONS 方法，但可以使用配置指令、自定义模块或 Lua 脚本实现。

02 超文本咖啡壶控制协议 HTCPCP 还有一个后续，HTCPCP-TEA（RFC7168），它用来控制茶壶。


**精选留言：**

- 壹笙☞漂泊 2019-06-19 10:19:25

  答题：

  1、增：POST 删：DELETE 改：PUT 查：GET

  请求头如何设计，这个问题。。。不太明白。

  2、我认为可以，因为http协议具有很好的灵活性。具体的对资源操作是由服务器决定的。

  总结：

  Http/1.1规定了八种方法，单词必须都是大写的形式。

  1. GET:获取资源，可以理解为读取或者下载数据
  2. HEAD:获取资源的元信息;
  3. POST:向资源提交数据，相当于写入或上传数据;
  4. PUT:类似POST;
  5. DELETE:删除资源;
  6. CONNECT:建立特殊的连接隧道;
  7. OPTIONS:列出可对资源实行的方法;
  8. TRACE:追踪请求-响应的传输路径。

  GET/HEAD

  ——从服务器获取资源

  HEAD和GET类似，也是从服务器获取资源，但是不会返回请求的实体数据，只有响应头（元信息），是GET的简易版，如果不需要资源的话，可以避免传输body数据的浪费。

  POST/PUT

  ——向服务器提交数据，数据在body里

  PUT和POST作用类似，有微妙不同，通常POST标识新建，PUT标识修改

  DELETE

  ——删除资源，危险性大，很少用

  CONNECT

——要求服务器为客户端和另一台远程服务器建立一条特殊的链接，这时Web服务器充当代理的角色

OPTIONS
——要求服务器列出可对资源实行的操作方法，在响应头Allow字段里返回。功能有限，用处不大。Nginx没支持

TRACE
——用于对HTTP链路的测试或诊断，可以显示出请求 - 响应的传输路径。存在漏洞，会泄露网站的信息，所以通常也是禁止使用

安全与幂等
安全：在HTTP协议里，所谓的安全，是指请求方法不会对服务器上的资源造成实质的修改，so 只有GET和HEAD是安全的，因为是只读操作。
幂等：多次执行相同的操作，结果也都是相同的。so GET和HEAD 即是安全的也是幂等的，DELETE可以多次删除同一个资源，效果都是“资源不存在”，所以也是幂等。POST是新增或提交数据，多次提交会创建多个资源，所以不是幂等的。PUT是替换或更新数据，多次更新一个资源，资源还是第一次更新的状态。所以是幂等的。
幂等：GET、HEAD、DELETE、PUT
非幂等：POST

[2赞]

作者回复2019-06-19 11:02:51
总结的非常好。
问题里的“请求头如何设计”，意思是说相关的curd参数应该放在什么地方，比如用query参数或者是字段，只是一个提示，不是要必须如何如何做。

- 大小兵 2019-06-19 01:13:33
真希望快点更新啊，看的不过瘾！ [2赞]

作者回复2019-06-19 09:00:38
慢慢来。

- 业余爱好者 2019-06-19 00:54:09
之前做一个网站的cms,觉得又是一套crud,毫无新意，闲得慌，于是玩了一波restful"架构"。严格按照http规范，比如，查询都用GET，新增用POST，更新用PUT，删除用DELETE，url的设计也按照rest风格设计。现在想想，tomcat支持这几种http方法也是万幸，不然的话，又得加班换成get/post了。

这段经历我认识到，http只是一种协议，不同的服务器，还有客户端，比如浏览器都可以有自己的实现。虽然各自在实现上有所取舍，但大体上，按照协议规范来，不会差。

协议，是个好东西。。 [2赞]

作者回复2019-06-19 09:02:02
请求方法的设计思想非常好，动词可以表示各种操作，所以非常适合RESTful。

- 一步 2019-06-19 21:14:06
老师 WebDav 这一块会详细讲嘛？ 有这块的需求，或者有没有好的文档 [1赞]

作者回复2019-06-20 08:55:44
这块我基本没用过，它也不是http标准里面的，抱歉了。

- 10 2019-06-19 17:01:25
  我采用POST /10-2 HTTP/1.1的指令写了10-2的内容为“POST DATA IS HERE”，然后我采用GET /10-2 HTTP/1.1的指令去读内容 返回的“200 OK”，但实体body的内容只是一个“0”，而非前面写的“POST DATA IS HERE”
  请问难道我前面的POST指令没有写成功么？ [1赞]

  作者回复2019-06-19 17:40:16
  测试用的URI“10-2”不支持存储数据，所以post的数据只能在当次请求生效。

  另外发现这两测试uri有小bug，已经修复，请及时git pull更新。

- 许童童 2019-06-19 11:27:02
  请求头里面应该要包含 请求的目标
  也就是对应数据库里面行 [1赞]

- 一步 2019-06-19 21:00:04
  OPTIONS 方法还是用的很多的，CORS跨域请求必须用到OPTIONS方法了

  作者回复2019-06-20 08:55:10
  我接触的领域里options用的比较少，可能有点孤陋寡闻了。

- 趙衍 2019-06-19 10:44:00
  关于Post和Get之间的区别，我一直很困惑，因为其实我们也可以在Get的请求体里写参数，用Get去修改资源；或者在Post的请求头上去写参数，用它去获取资源。所以他们两者之间到底有什么区别呢？希望老师可以指教一下！

  作者回复2019-06-19 11:00:41
  就是个使用的习惯和约定，就像是红绿灯，不是强制要求你必须遵守，但大家都按照这样做沟通起来顺畅。

  要理解协议的含义，要求你遵守，但不强制。

- 彧豪 2019-06-19 10:37:42
  老师，话说我昨天实际项目终于到一个问题：
  get请求带上查询字符串例如?name=a+b，但是打开chrome的控制台network选项卡发现请求url那是对的，是xxx?name=a+b，但是最后的查询字符串那那里是name: a b，+号变成了空格，java那边收到的也是a b……
  最后的解决方法是我这边encodeURIComponent一下，java那边URLDecoder.decode一下即可
  但是我不明白为何会如此，为何get请求的查询字符串中带+号，浏览器会将其变为空格
  在浏览器控制台network选项卡底端的query string parameters那有个"view URL encoded"按钮，点了之后name:a b变为name:a+b，这就意味着空格被转义成了+号，那是不是这个a和b之间的字符(串)实际不是空格呢？毕竟我暂时想不到什么方法能将空格转义为+号
  关于+号变空格的问题希望老师能指点迷津，感谢

  作者回复2019-06-19 12:16:08
  下一讲里面的url编码就会谈到。
  uri里面有些字符是不允许出现的，需要编码和解码，而+正好就被解码成了空格。

  你刚才也说了，会用到encodeURI这样的函数。

- 汤小高 2019-06-19 10:01:28
  老师，请问：幂等性指多次执行相同的操作，获取的结果是相同的。但是，比如get获取某个服务器资源，某个时刻获取的资源和下一时刻获取的资源也可能不一样吧，比如下一时刻服务器对该资源进行了改变，再get时，客户端获取的不就不一样了吗？

  作者回复2019-06-19 11:04:36
  幂等是指客户端操作对服务器的状态没有产生改变，虽然报文内容变了，但服务器还是没有变化。

  可以再对比一下delete和post。

- 1900 2019-06-19 09:43:29
  “幂等”有什么具体的落地场景么？它重要的原因在哪里呢？

  作者回复2019-06-19 11:05:41
  在RESTful设计的时候，要考虑动作对服务器内部状态的影响。

  这个比较理论，一般不太需要关心。

- -W.LI- 2019-06-19 09:42:35
  老师好!我项目用得nginx好像支持options请求，put请求幂等这个是不是有待商榷啊

  作者回复2019-06-19 11:07:26
  我看Nginx源码，是不支持options的。

  put的幂等是RFC定义的，当然你要实现出非幂等的功能也是可以的。

- MJ 2019-06-19 09:39:00
  有没有同学遇到同样的问题，虚拟机搭建的实验环境，telnet没法操作，输入一个字符，就报HTTP/1.1 400 bad request.

- 一粟 2019-06-19 06:46:05
  访问某一网站，只有对方的IP和端口号，不知道index在的资源路径，有办法寻找到么?

  作者回复2019-06-19 09:01:17
  这个不太清楚，或许要做什么“扫描”？

- 浩浩 2019-06-19 01:04:45
  老师有个问题想问一下，我之前用比较旧的http工具类发请求，使用get方法时不能使用请求体，但使用软件工具能使用请求体，不同方法对请求的格式要求是不是一样，还有就是不同方法的区别是不是体现在服务器的响应上，用post请求访问一张照片会是什么样的，暂时还没试过

  作者回复2019-06-19 09:07:05
  任何请求报文都可以带请求体，与方法无关，老的工具可能对协议支持的不好。

  后面你理解的对，请求方法最终如何处理还是要依赖于服务器，如果愿意，服务器也可以用get来上传数据，用post获取数据，现在的get/post都是依据协议标准来执行，所以是get获取post上传。