HTTP

# http状态码

1xx 信息

|  |  |
| --- | --- |
| 消息 | 描述 |
| 100 continue | 服务器仅接收到部分请求，但是一旦服务器并没有拒绝该请求，客户端应该继续发送其余的请求。 |
| 101 switching protocols | 服务器转换协议:服务器将遵从客户的请求转换到另一种协议 |

2xx 信息

|  |  |
| --- | --- |
| 消息 | 描述 |
| 200 OK | 请求成功(其后对GET或POST请求的应答文档) |
| 201 Created | 请求被创建完成，同时新的资源被创建 |
| 202 Accepted | 供处理的请求已被接受，但是处理未完成 |
| 203 Non-authoritative Information | 文档已经正常地返回，但一些应答头可能不正确，因为使用的是文档的拷贝 |
| 204 no content | 没有新文档。浏览器应该继续显示原来的文档。如果用户定期地刷新页面，而Servlet可以确定用户文档足够新，这个状态代码是很有用的。 |
| 205 reset content | 没有新文档。但浏览器应该重置它所显示的内容。用来强制浏览器清除表单输入内容 |
| 206 Partial Content | 客户发送了一个带有Range头的GET请求，服务器完成了它 |

3xx

|  |  |
| --- | --- |
| 消息 | 描述 |
| 300 Multiple Choices | 多重选择。链接列表。用户可以选择某链接到达目的地。最多允许五个地址 |
| 301 Moved Permanently | 所请求的页面已经转移至新的url |
| 302 Found | 锁清秋的页面已经临时转移至新的url |
| 303 See Other | 所请求的页面可在别的url下被找到 |
| 304 Not Modified | 未按预期修改文档。客户端有缓冲的文档并发送了一个条件性的请求(一般是提供if-modified-since头表示客户只想比指定日期更新的文档)。服务器告诉客户，原来缓冲的文档还可以继续使用。 |
| 305 Use Proxy | 客户请求的文档应该通过Location头所指明的代理服务器提取 |
| 306 Unused | 此代码被用于前一版本。目前已不再使用，但是代码依然被保留 |
| 307 Temporary Redirect | 被请求的页面已经临时移至新的url |

4xx

|  |  |
| --- | --- |
| 消息 | 描述 |
| 400 Bad Request | 服务器未能理解请求 |
| 401 Unauthorized | 被请求的页面需要用户名和密码 |
| 402 Payment Required | 此代码尚无法使用 |
| 403 Forbidden | 对被请求页面的访问被禁止 |
| 404 Not Found | 服务器无法找到被请求的页面 |
| 405 Method Not Allowed | 请求中指定的方法不被允许 |
| 406 Not Acceptable | 服务器生成的响应无法被客户端所接受 |
| 407 Proxy Authentication Required | 用户必须首先使用代理服务器进行验证，这样请求才会被处理 |
| 408 Request Timeout | 请求超出了服务器的等待时间 |
| 409 Conflict | 由于冲突，请求无法被完成 |
| 410 Gone | 被请求的页面不可用 |
| 411 Length Required | “Content-Length”未被定义。如果无此内容，服务器不会接受请求 |
| 412 Precondition Failed | 请求中的前提条件被服务器评估为失败 |
| 413 Request Entity Too Large | 由于所请求的实体的太大，服务器不会接受请求 |
| 414 Request-url Too Long | 由于url太长，服务器不会接受请求。当post请求被转换为带有很长的查询信息的get请求时，就会发生这种请求 |
| 415 Unsupported Media Type | 由于媒介类型不被支持，服务器不会接受请求 |
| 416 | 服务器不能满足客户在请求中指定的Range头 |
| 417 Expectation Failed |  |

5xx

|  |  |
| --- | --- |
| 消息 | 描述 |
| 500 Internal Server Error | 请求问完成。服务器遇到不可预知的情况 |
| 501 Not Implemented | 请求未完成。服务器不支持所请求的功能 |
| 502 Bad Gateway | 请求为完成。服务器从上游服务器收到一个无效的响应 |
| 503 Service Unavailable | 请求未完成。服务器临时过载或宕机 |
| 504 Gateway Timeout | 网关超时 |
| 505 HTTP Version Not Support | 服务器不支持请求中指明的HTTP协议版本 |

# 什么是HTTP

超文本传输协议(HTTP)的设计目的是保证客户机与服务器之间的通信。

HTTP的工作方式是客户机与服务器之间的请求-应答协议。

web浏览器可能是客户端，而计算机上的网络应用程序也可能作为服务器端。

举例：客户端(浏览器)向服务器提交HTTP请求；服务器向客户端返回响应。响应包含关于请求的状态信息以及可能被请求的内容。

# HTTP请求方法 GET和POST

在客户机和服务器之间进行请求-响应时，两种最常用的方法是：GET和POST。

▪ GET – 从指定的资源请求数据；

▪ POST – 向指定的资源提交要被处理的数据。

## GET

查询字符串(名称/值)是在GET请求的URL中发送的，例如：

|  |
| --- |
| /test/demo\_form.asp?name1=value1&name2=value2 |

▪ GET 请求可被缓存；

▪ GET 请求保留在浏览器历史记录中；

▪ GET 请求可被收藏为书签；

▪ GET 请求不应该在处理敏感数据时使用；

▪ GET 请求有长度限制；

▪ GET 请求只应当用于取数据。

## POST

查询字符串(名称/值)是在POST请求的HTTP消息主题中发送的，例如：

|  |
| --- |
| POST /test/demo\_form.asp HTTP/1.1  Host: w3schools.com  name1=value1&name2=value2 |

▪ POST请求不会被缓存；

▪ POST请求不会保留在浏览器历史记录中；

▪ POST 不能被收藏为书签；

▪ POST请求对数据长度没有要求。

## 比较 GET 与 POST

下面的表格比较了两种 HTTP 方法：GET 和 POST。

|  |  |  |
| --- | --- | --- |
|  | **GET** | **POST** |
| 后退按钮/刷新 | 无害 | 数据会被重新提交（浏览器应该告知用户数据会被重新提交）。 |
| 书签 | 可收藏为书签 | 不可收藏为书签 |
| 缓存 | 能被缓存 | 不能缓存 |
| 编码类型 | application/x-www-form-urlencoded | application/x-www-form-urlencoded 或 multipart/form-data。为二进制数据使用多重编码。 |
| 历史 | 参数保留在浏览器历史中。 | 参数不会保存在浏览器历史中。 |
| 对数据长度的限制 | 是的。当发送数据时，GET 方法向 URL 添加数据；URL 的长度是受限制的（URL 的最大长度是 2048 个字符）。 | 无限制。 |
| 对数据类型的限制 | 只允许 ASCII 字符。 | 没有限制。也允许二进制数据。 |
| 安全性 | 与 POST 相比，GET 的安全性较差，因为所发送的数据是 URL 的一部分。  在发送密码或其他敏感信息时绝不要使用 GET ！ | POST 比 GET 更安全，因为参数不会被保存在浏览器历史或 web 服务器日志中。 |
| 可见性 | 数据在 URL 中对所有人都是可见的。 | 数据不会显示在 URL 中。 |

# 其他 HTTP 请求方法

下面的表格列出了其他一些 HTTP 请求方法：

|  |  |
| --- | --- |
| **方法** | **描述** |
| HEAD | 与 GET 相同，但只返回 HTTP 报头，不返回文档主体。 |
| PUT | 上传指定的 URI 表示。 |
| DELETE | 删除指定资源。 |
| OPTIONS | 返回服务器支持的 HTTP 方法。 |
| CONNECT | 把请求连接转换到透明的 TCP/IP 通道。 |

# GET和POST本质上没有区别

get和post是什么，http协议中的两种发送请求的方法。

http是什么？http是基于tcp/ip的关于数据如何在万维网中如何通信的协议。

http的底层是tcp/ip。所以get和post的底层也是tcp/ip，也就是说，get/post都是tcp链接。get和post能做的事情是一样的。你要给get加上request body，给post带上url参数，技术上是完全行得通的。

那么，“标准答案”里那些区别是怎么回事？

在我大万维网世界中，TCP就像汽车，我们用TCP来运输数据，它很可靠，从来不会发生丢件少件的现象。但是如果路上跑的全是看起来一模一样的汽车，那这个世界看起来是一团混乱，送急件的汽车可能被前面满载货物的汽车拦堵在路上，整个交通系统一定会瘫痪。为了避免这种情况发生，交通规则HTTP诞生了。HTTP给汽车运输设定了好几个服务类别，有GET, POST, PUT, DELETE等等，HTTP规定，当执行GET请求的时候，要给汽车贴上GET的标签(设置method为GET)，而且要求把传送的数据放在车顶上(url中)以方便记录。如果是POST请求，就要在车上贴上POST的标签，并把货物放在车厢里。当然，你也可以在GET的时候往车厢内偷偷藏点货物，但是这是很不光彩;也可以在POST的时候在车顶上也放一些数据，让人觉得傻乎乎的。HTTP只是个行为准则，而TCP才是GET和POST怎么实现的基本。

但是，我们只看到http对get和post参数的传送渠道(url还是request body)提出了要求，“标准答案”里关于参数大小的限制又是从哪来的呢？

在万维网世界中，还有另一个重要的角色：运输公司。不同的浏览器(发起http请求)和服务器(接受http请求)就是不同的运输公司。虽然理论上，你可以在车顶上无限的堆货物(url中无限加参数)。但是运输公司可不傻，装载和卸载也是有很大成本的，他们会限制单词运输量来控制风险，数据量太大对浏览器和服务器都是很大的负担。业界不成文的规定是，(大多数)浏览器通常都会限制url长度在2K个字节，而(大多数)服务器最多处理64K大小的url。超过的部分，恕不处理。如果你用get服务，在request body偷偷藏了数据，不同服务器的处理方式也是不同的，有些服务器会帮你卸载，读出数据，有些服务器直接忽略，所以，虽然get可以带request body，也不能保证一定能被接受到。

好了，现在你知道，get和post本质上就是tcp链接，并无差别。但是由于http的规定和浏览器/服务器的限制，导致它们在应用过程中体现出一些不同。

get和post有一个重大的区别，简单来说：

get产生一个tcp数据包；

post产生两个tcp数据包。

长话来说：

对于get方式的请求，浏览器会把httpheader和data一并发送出去，服务器响应200(返回数据)

对于post，浏览器先发送header，服务器响应100continue，浏览器再发送data服务器响应200 OK(返回数据)

也就是说，get值需要汽车跑一趟就把货送到了，而post要跑两趟，第一趟，先去和服务器大哥招呼，“Hi”，我等下要发送一批货来，你们打开门迎接我，然后再回头把货送过去。

因为post需要两部，舌尖上消耗的要多一点，看起来get比post更高效。因此Yahoo团队有推荐get替换post来优化网站性能。但这是一个坑！跳入需谨慎。为什么？

▪ get与post都有自己的寓意，不能随便混用；

▪ 据研究，在网络环境好的情况下，发送一次包的时间和发送两次包的时间差基本可以忽略。而在网络环境差的情况下，两次包的tcp在验证数据包完整性上，有非常大的优点。

▪ 并不是所有浏览器都会在post中发送两次包，Firefox就只发送一次。