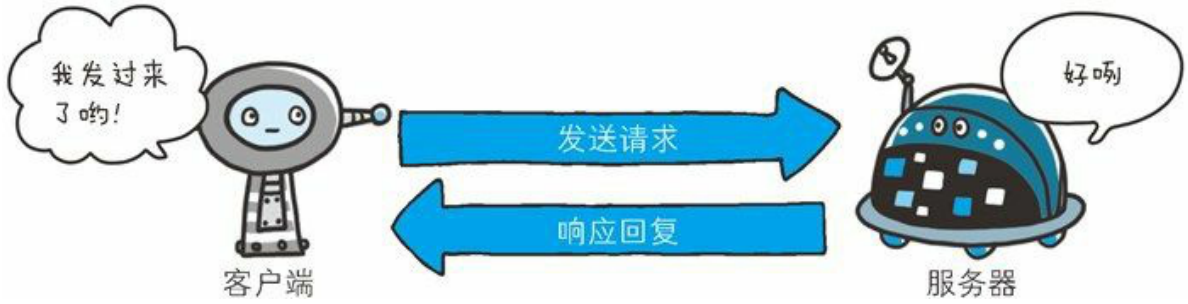
HTTP详解

一、定义

HTTP（Hyper Text Transfer Protocol）： 全称超文本传输协议，是用于从 Web 服务器传输超文本到本地浏览器的传送协议。通俗点讲，协议就是要保证网络通信的双方，能够互相对接上号。

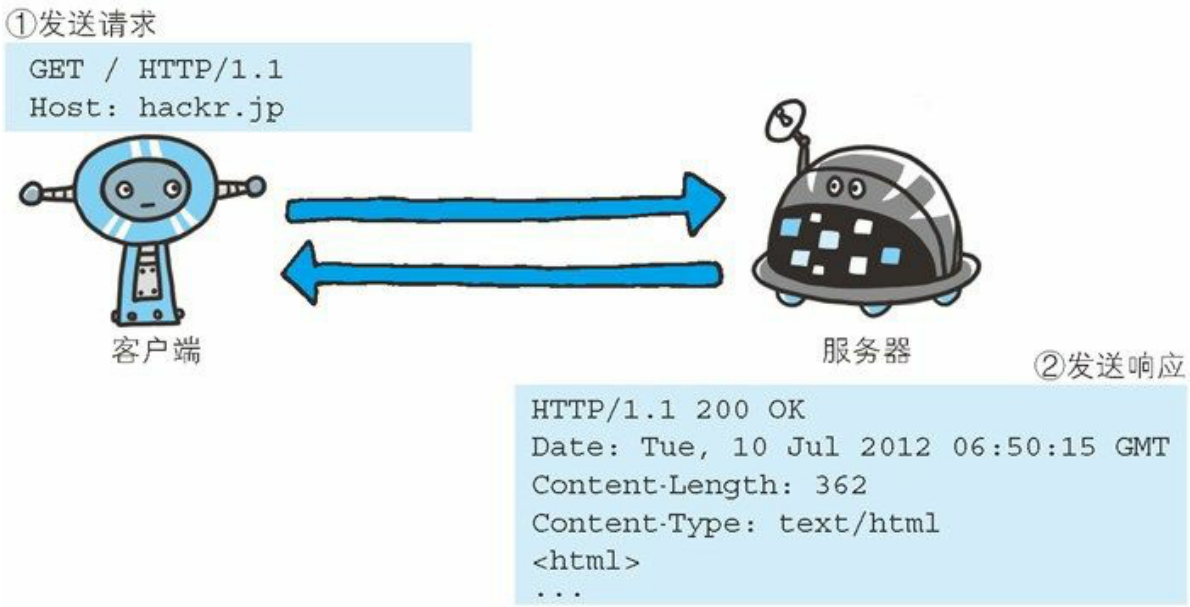
HTTP协议是一个基于请求与响应模式的、无连接、无状态的应用层协议，它是基于TCP/IP之上的一种连接方式，HTTP协议1.1以上版本提供一种长连接机制。

1、请求和响应模式



HTTP协议规定,请求从客户端发出,最后服务器端响应该请求并返回。换句话说,肯定是先从客户端开始建立通信的，服务器端在没有接收到请求之前不会发送响应 。

具体示例



下面则是从客户端发送给某个 HTTP 服务器端的请求报文中的内容。

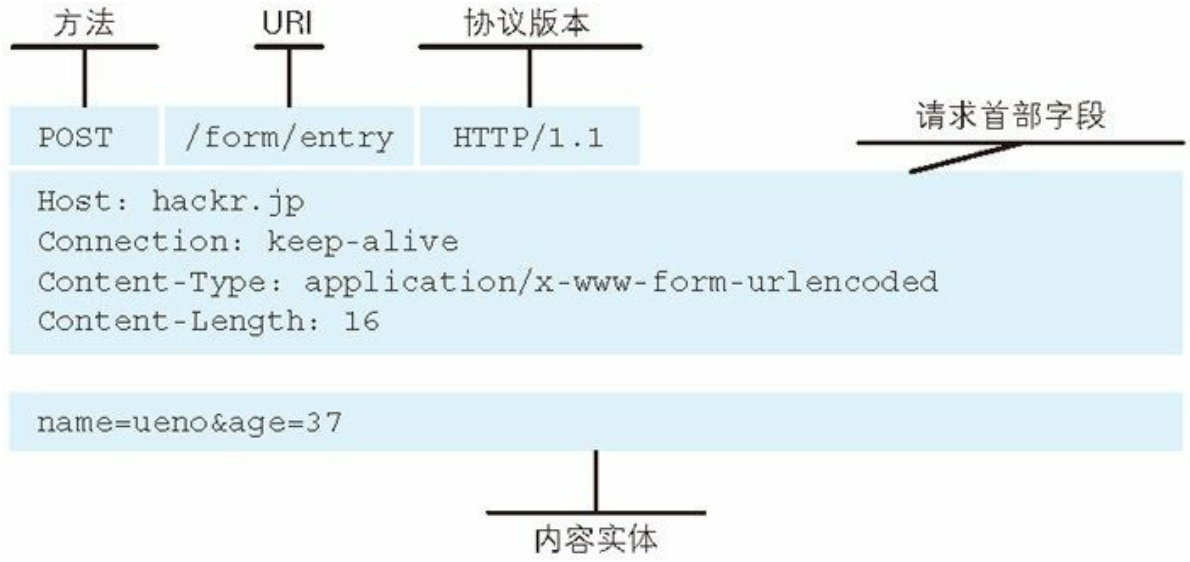
```
GET /index.htm HTTP/1.1
Host: hackr.jp
```

起始行开头的GET表示请求访问服务器的类型，称为请求方法。随后的字符串 /index.htm 指明了请求访问的资源对象，也叫做请求 URI。最后的 HTTP/1.1，即 HTTP 的版本 号，用来提示客户端使用的 HTTP 协议功能。

综合来看，这段请求内容的意思是：请求访问某台 HTTP 服务器上的 /index.htm 页面资源。

请求报文是由请求方法、请求 URI、协议版本、可选的请求头和请求体构成的。

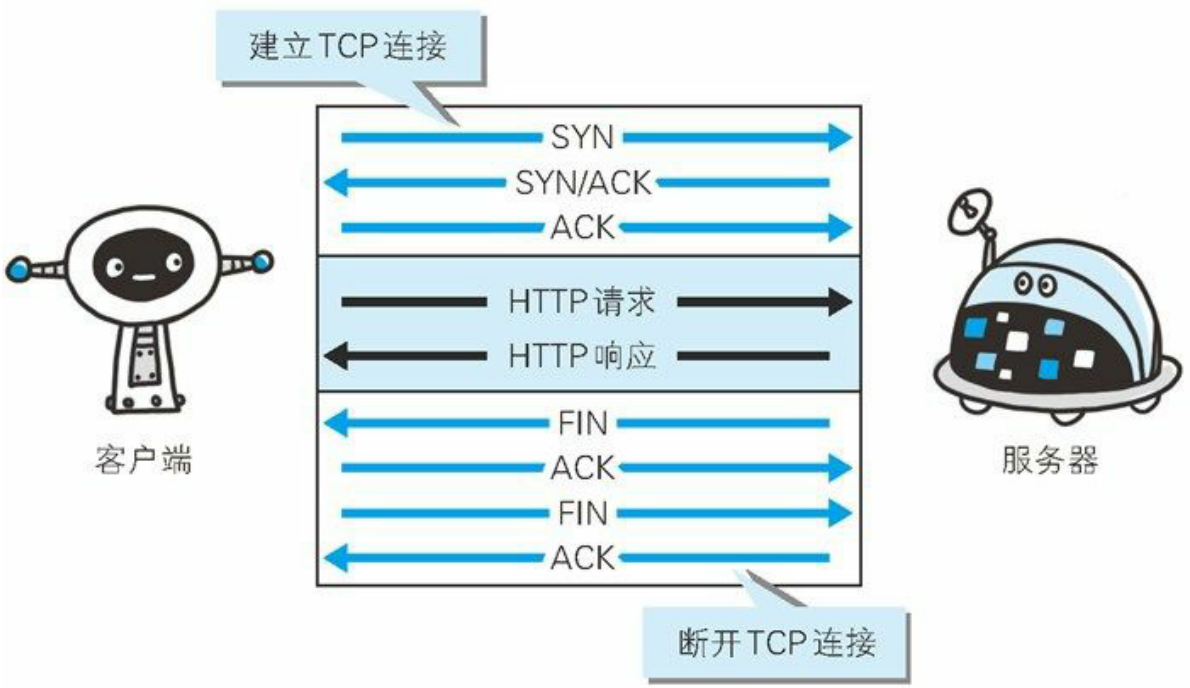
请求报文构成图例



响应报文构成图例

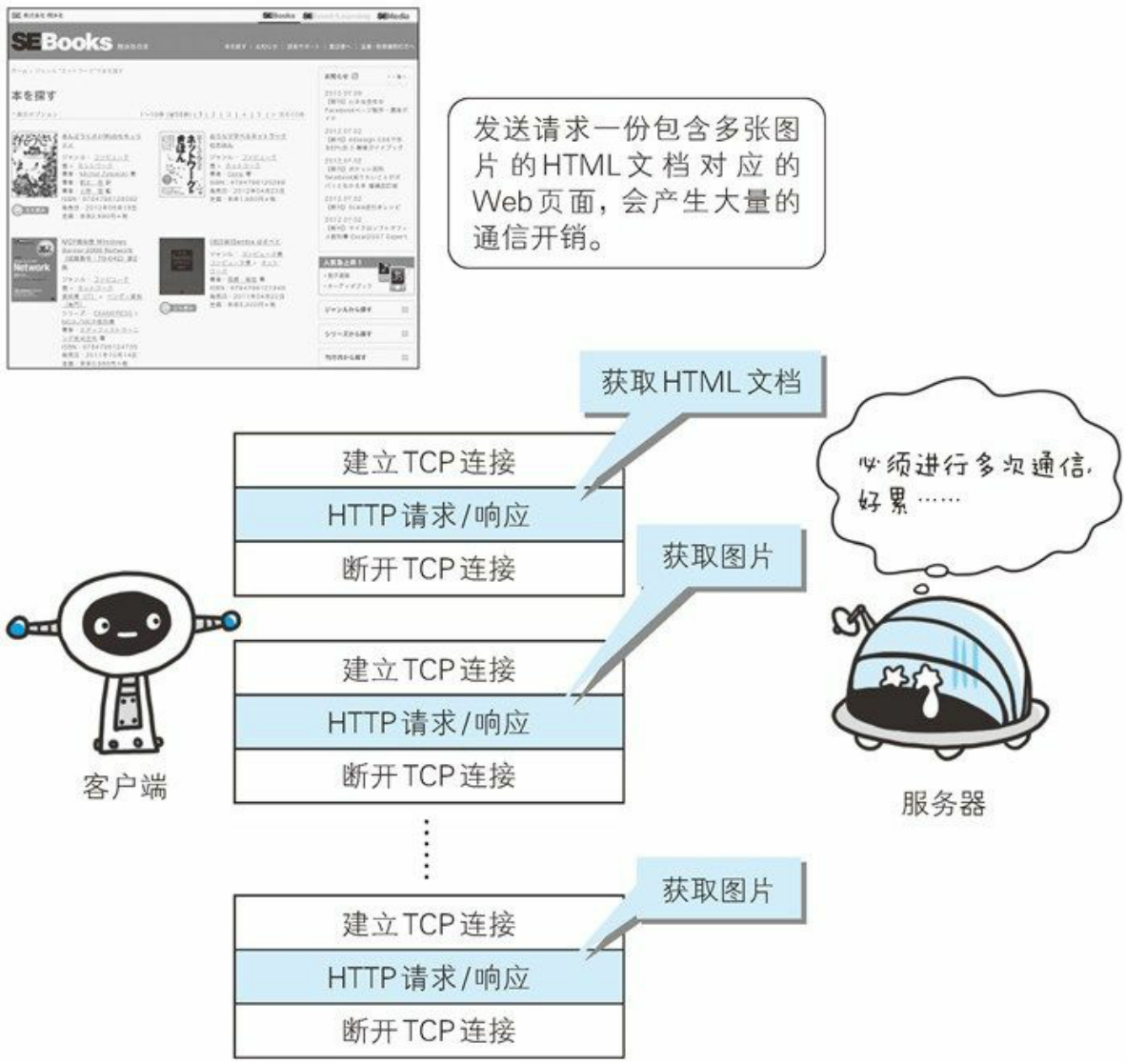
2、无连接

无连接的含义是限制每次连接只处理一个请求。服务器处理完客户的请求，并收到客户的应答后，即断开连接。采用这种方式可以节省传输时间

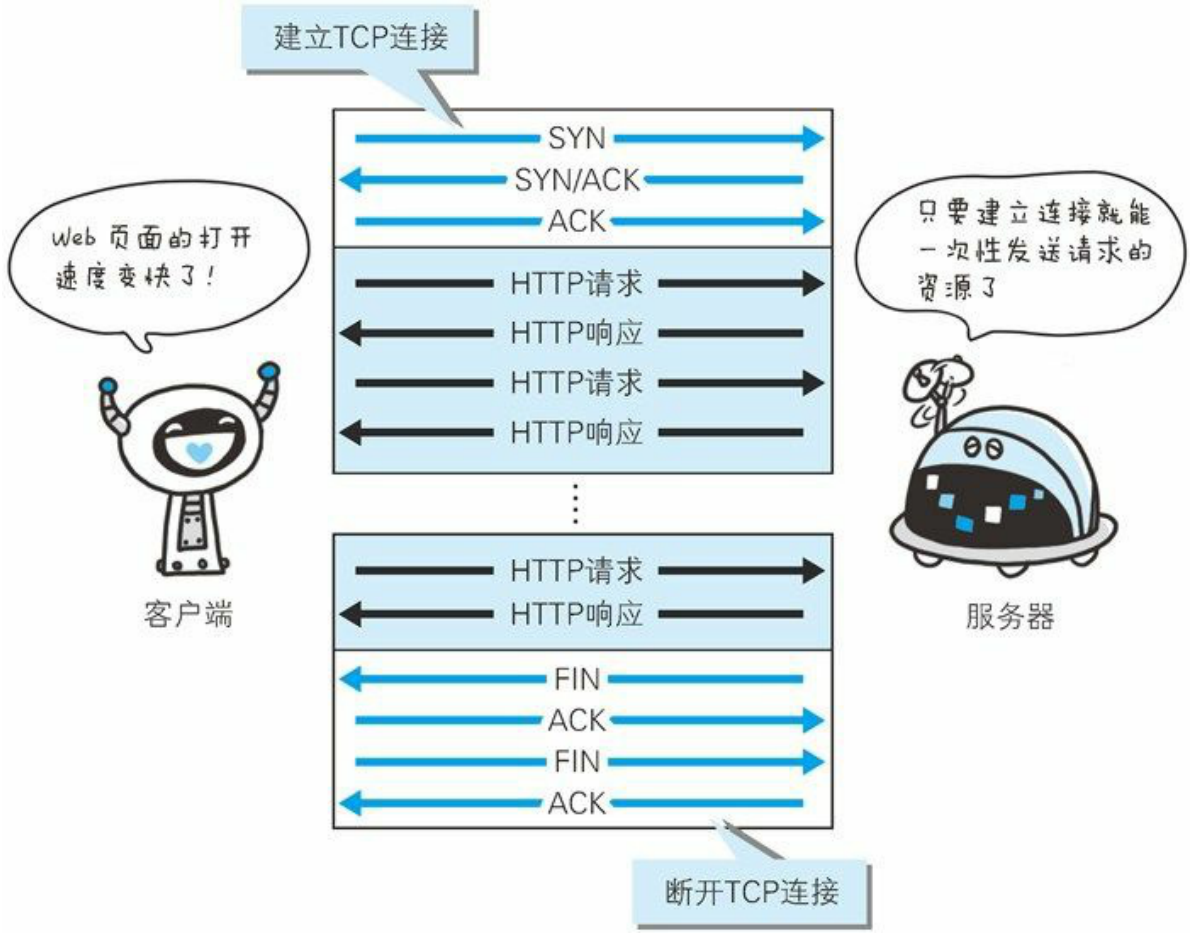


早期这么做的原因是 HTTP 协议产生于互联网，因此服务器需要处理同时面向全世界数十万、上百万客户端的网页访问，但每个客户端（即浏览器）与服务器之间交换数据的间歇性较大（即传输具有突发性、瞬时性），并且网页浏览的联想性、发散性导致两次传送的数据关联性很低，大部分通道实际上会很空闲、无端占用资源。因此 HTTP 的设计者有意利用这种特点将协议设计为**请求时建连接、请求完释放连接，以尽快将资源释放出来服务其他客户端。**

随着时间的推移，网页变得越来越复杂，里面可能嵌入了很多图片，这时候每次访问图片都需要建立一次 TCP 连接就显得很低效。



为解决上述 TCP 连接的问题，**HTTP/1.1 想出了持久连接（keep-alive）的方法。持久连接的特点是，只要任意一端没有明确提出断开连接，则保持 TCP 连接状态。**



持久连接的好处在于减少了 TCP 连接的重复建立和断开所造成的额外开销，减轻了服务器端的负载。另外，减少开销的那部分时间，使HTTP 请求和响应能够更早地结束，这样 Web 页面的显示速度也就相应提高了。

2、无状态

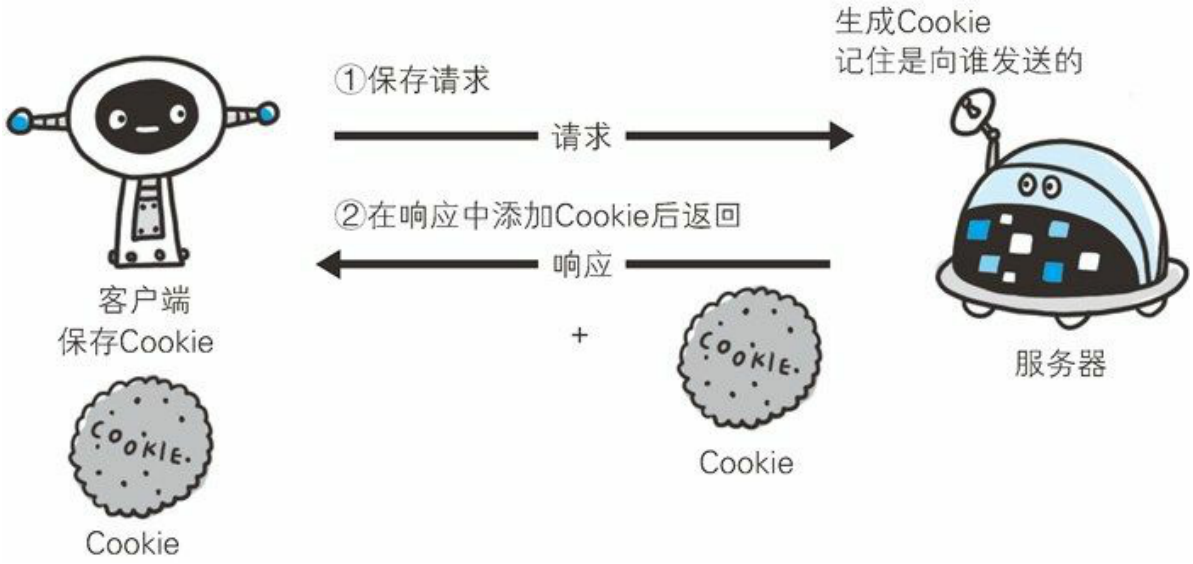
HTTP 是一种不保存状态，即无状态（stateless）协议。HTTP 协议自身不对请求和响应之间的通信状态进行保存。也就是说在 HTTP 协议对于发送过的请求或响应都不做持久化处理。

使用HTTP协议，一次请求只会对应一次响应，且不会记录之前的所有请求响应信息。可是，随着 Web 的不断发展，因无状态而导致业务处理变得棘手的情况增多了。比如，用户登录到一家购物网站，即使他跳转到该站的其他页面后，也需要能继续保持登录状态。针对这个实例，网站为了能够掌握是谁送出的请求，需要保存用户的状态，于是引入了Cookie技术。有了Cookie再用HTTP协议通信，就可以管理状态了。

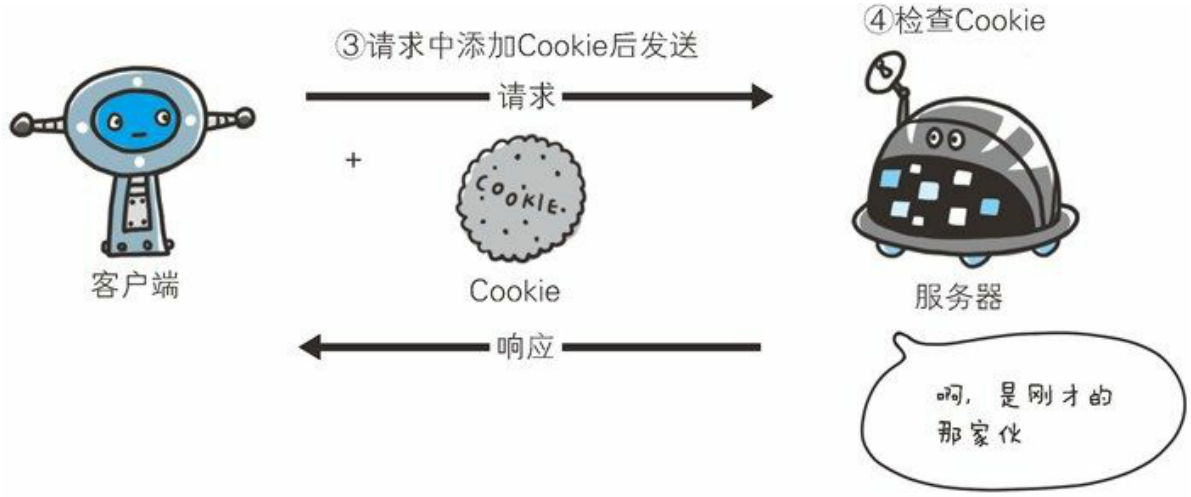
3、Cookie

Cookie 技术通过在请求和响应报文中写入 Cookie 信息来控制客户端的状态。 Cookie 会根据从服务器端发送的响应报文内的一个叫做 Set-Cookie 的请求头，通知客户端保存 Cookie。当下次客户端再往该服务器发送请求时，客户端会自动在请求报文中加入 Cookie 值后发送出去。并且服务器还可以根据需要修改Cookie的内容。

示例：没有 Cookie 信息状态下的请求

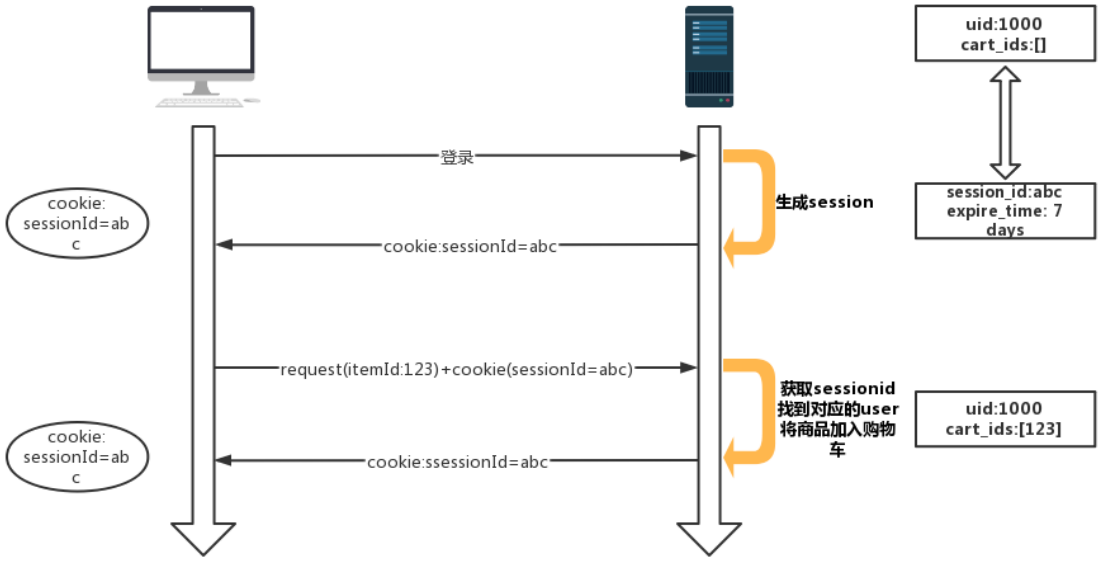


示例：第 2 次以后（存有 Cookie 信息状态）的请求



4、Session

Session是另一种记录客户状态的机制，不同的是Cookie保存在客户端浏览器中，而Session保存在服务器上。客户端浏览器访问服务器的时候，服务器把客户端信息以某种形式记录在服务器上。这就是Session。客户端浏览器再次访问时只需要从该Session中查找该客户的状态就可以了。



二、 HTTP请求流程

1、输入url，回车

URL 遵守一种标准的语法，它由协议、主机名、域名、端口、路径、文件名、参数这几个部分构成具体语法规则如下：

协议://主机名.域名:端口/路径/文件名?参数

2、地址解析 --- 域名转换成ip地址

3、 浏览器与WEB服务器建立一个TCP连接（三次握手）

4、 浏览器给WEB服务器发送一个HTTP请求

5、 服务器端响应HTTP请求，浏览器得到HTML代码

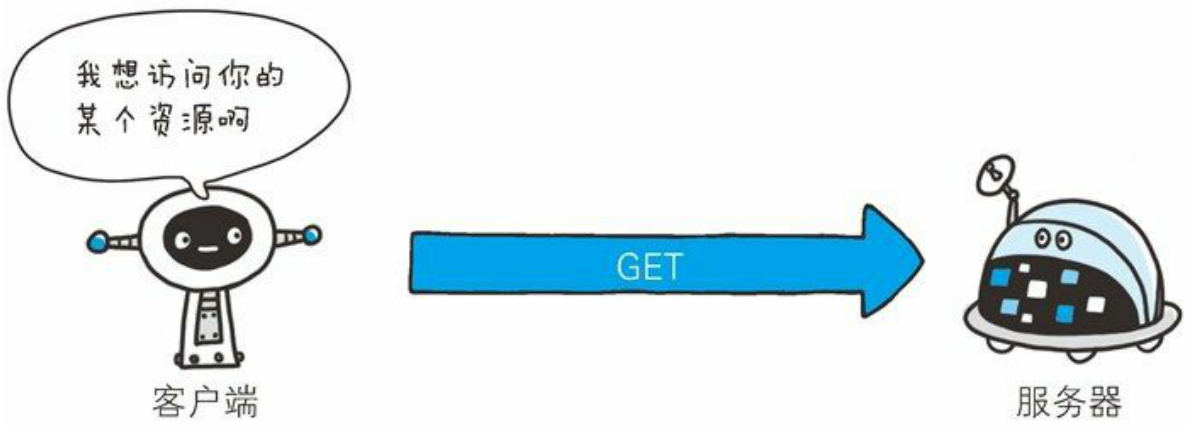
6、 浏览器解析HTML代码，并请求HTML代码中的资源

7、 关闭TCP连接，浏览器对页面进行渲染呈现给用户

三、 HTTP请求方法

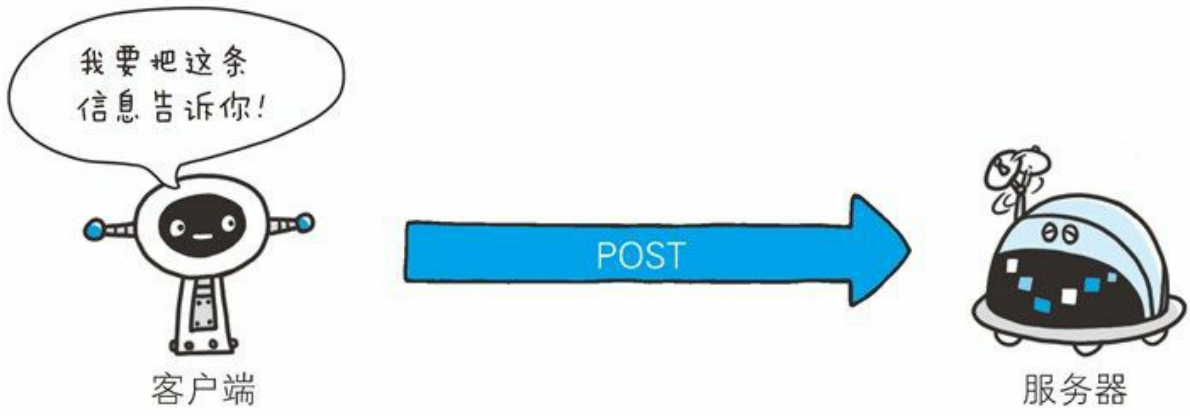
GET： 从指定的资源请求数据

用于请求服务器获取指定资源。这是最常见的的请求方法，在浏览器地址栏输入URL或点击超链接时，浏览器发送的都是GET请求。GET请求可以将参数附加在URL后面，参数之间使用`?`分隔，键值对之间使用`=`连接，多个参数使用`&`分隔。指定的资源经服务器端解析后返回响应内容。也就是说，如果请求的资源是静态文本，那就保持原样返回；如果是动态页面，则返回经过执行后的输出结果。



POST： 向指定的资源提交要被处理的数据。

用于向服务器提交数据，以供创建或更新资源。POST请求将数据放在请求体中发送，而不是URL中。与GET请求相比，POST请求可以传输更多数据，没有数据大小的限制，主要取决于服务器处理程序的能力。POST请求常用于提交表单数据或上传文件等。

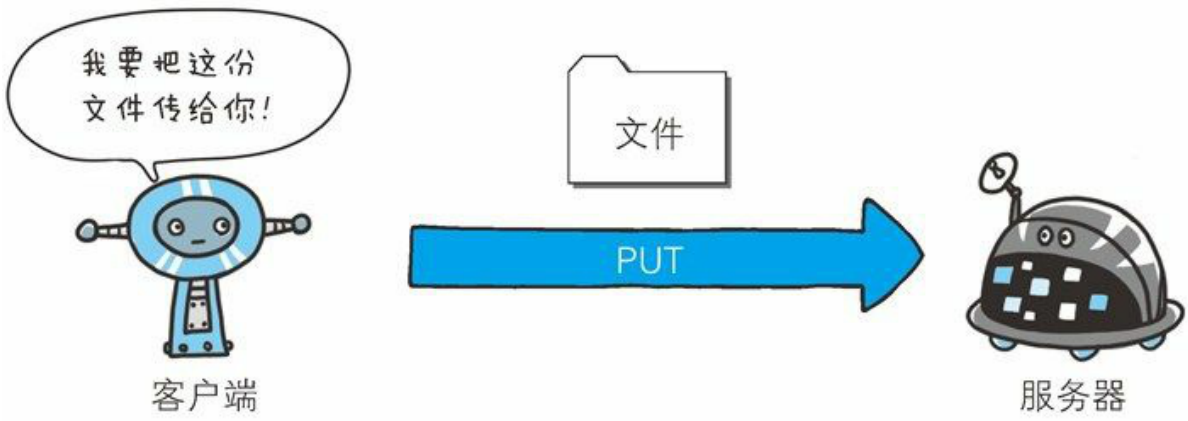


	GET	POST
后退按钮/刷新	无害	数据会被重新提交（浏览器应该告知用户数据会被重新提交）。
书签	可收藏为书签	不可收藏为书签
历史	参数保留在浏览器历史中。	参数不会保存在浏览器历史中。

	GET	POST
对数据长度的限制	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，数据在 URL 中对所有人都是可见的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，因为数据不会显示在 URL 中，且参数不会被保存在浏览器历史中。

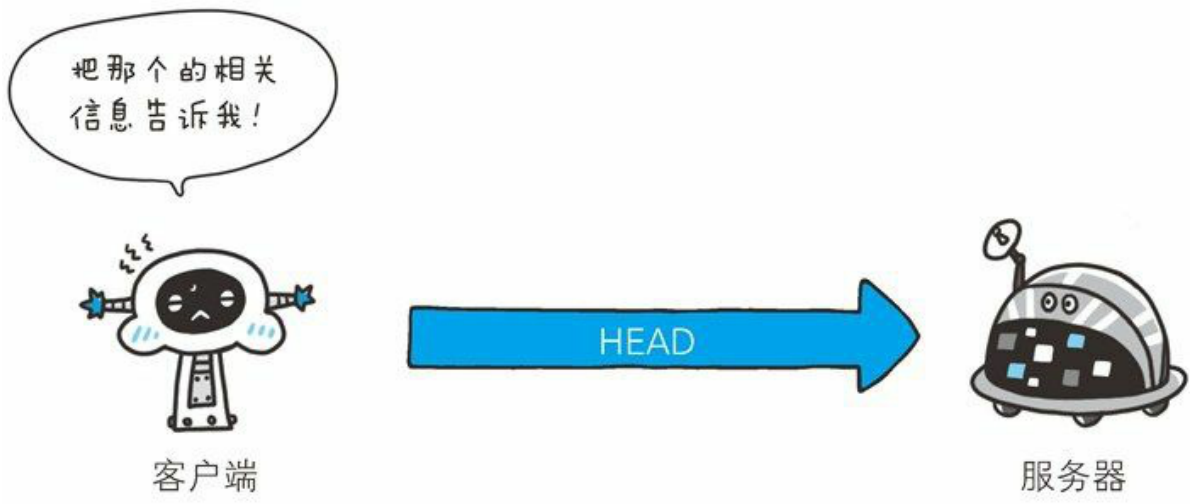
PUT：传输文件

PUT 方法用来传输文件。就像 FTP 协议的文件上传一样，要求在请求报文的请求体中包含文件内容，然后保存到请求 URI 指定的位置。但是，由于PUT 方法自身不带验证机制，任何人都可以上传文件，存在安全性问题，因此一般的 Web 网站不使用该方法。



HEAD：获得请求头

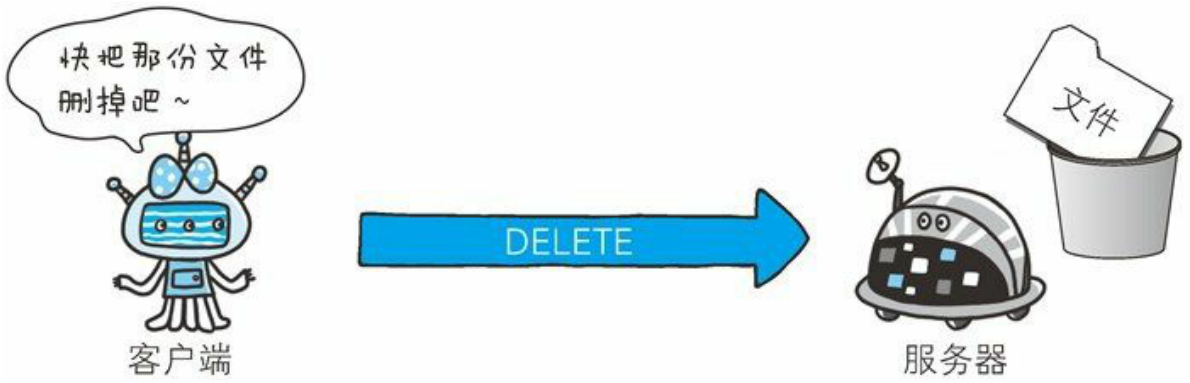
HEAD 方法和 GET 方法一样，只是不返回报文主体部分。用于确认资源的有效性及资源更新的日期时间等。



DELETE：删除文件

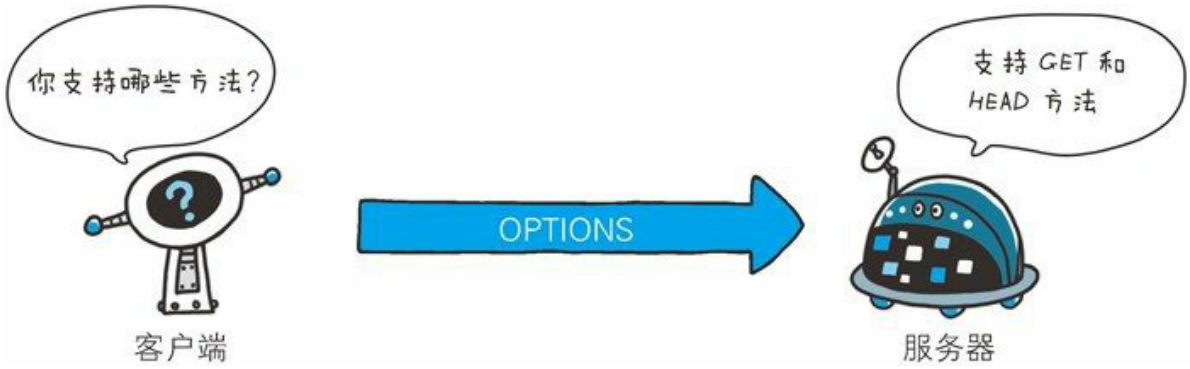
DELETE 方法用来删除文件，是与 PUT 相反的方法。DELETE 方法按请求 URI 删除指定的资源。

但是，DELETE 方法本身和 PUT 方法一样不带验证机制，所以一般的 Web 网站也不使用 DELETE 方法。



OPTIONS：询问支持的方法

OPTIONS 方法用来查询指定的资源支持的请求方法。

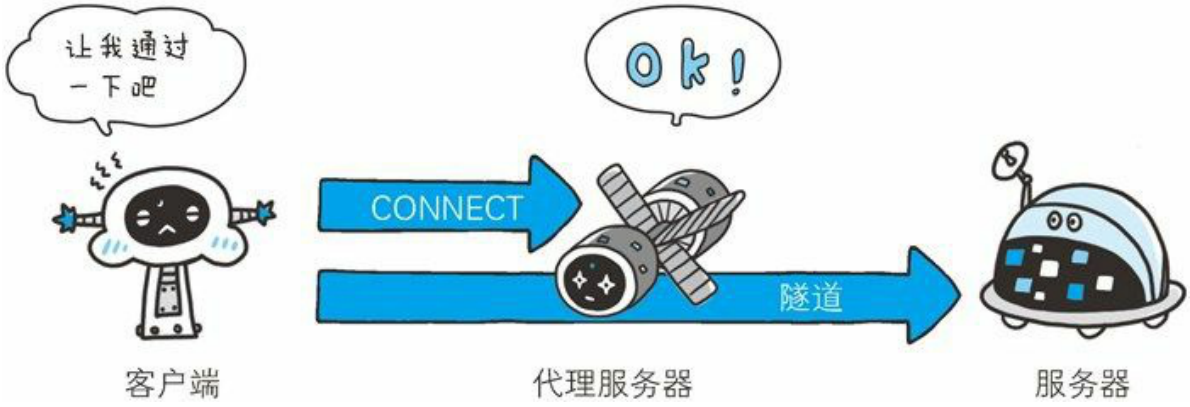


TRACE：回显服务器收到的请求，主要用于测试或诊断

客户端可以发送TRACE请求给服务器,服务器会将该请求的完整内容以响应的形式返回给客户端,这样客户端就可以看到服务器实际收到的请求信息。这可以用于测试或调试,确认请求是否如预期那样到达了服务器,或者请求中是否丢失了某些信息。

CONNECT：要求用隧道协议连接代理

CONNECT 方法要求在与代理服务器通信时建立隧道，实现用隧道协议进行 TCP 通信。



方法	说明	支持的 HTTP 协议版本
GET	获取资源	1.0、1.1
POST	提交数据	1.0、1.1
PUT	传输文件	1.0、1.1
HEAD	获得请求头	1.0、1.1
DELETE	删除文件	1.0、1.1
OPTIONS	询问支持的方法	1.1
TRACE	进行测试，查看服务器收到的报文信息	1.1
CONNECT	要求用隧道协议连接代理	1.1

OPTIONS请求的原因

产生OPTIONS请求的原因包括以下几条：

1：产生了复杂请求。

复杂请求对应的就是简单请求，不满足简单请求条件的就是复杂请求。

简单请求的定义是：

1. 请求方法是GET、HEAD或者POST， 并且当请求方法是 POST 时 Content-Type 请求头必须是下列三种之一。

   application/x-www-form-urlencoded（默认的表单提交数据格式，会将表单字段与其内容组合，转换为键值对序列，然后经过URL编码后提交到服务器。）

   multipart/form-data（这种格式会将表单的数据处理为一条消息，以标签为单元，用分隔符分开。常见于有文件上传的场景。）

   text/plain（表示纯文本格式，表单数据不进行编码，以纯文本方式进行提交。）

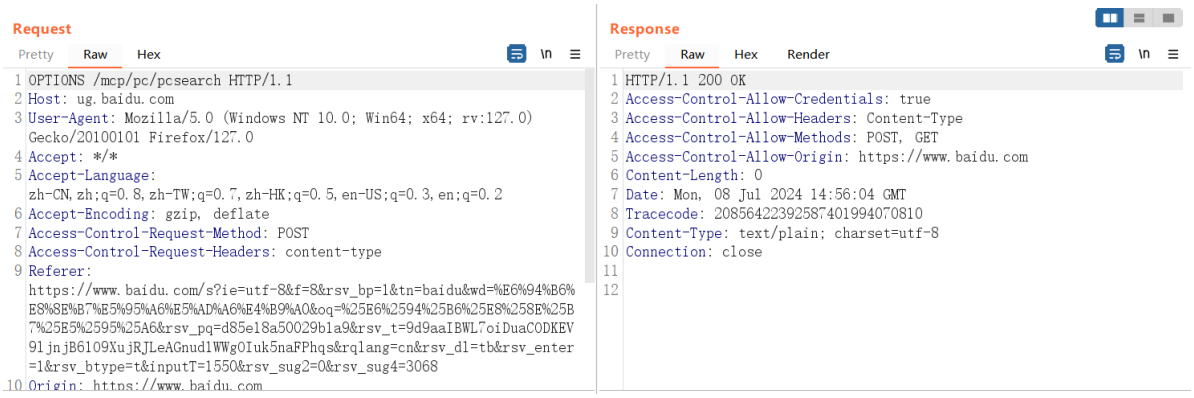
2. 请求中没有自定义HTTP头部。

2：发生了跨域。

要同时满足以下三个条件，只要有任何一个不同，都被当作是不同的域：

协议	域名	端口	是否同源	原因
http	www.a.com	80		
http	www.b.com	80	否	域名不同
https	www.a.com	80	否	协议不同
http	www.a.com	8080	否	端口不同

在进行跨域资源共享(CORS)时会使用 options 请求进行预检，以验证实际请求是否安全。跨域资源共享(Cross-Origin Resource Sharing，简称CORS)是一种浏览器技术，它允许在一个域名的网页请求其他域名的资源。



在OPTIONS请求中，通过请求头将 Access-Control-Request-Headers 与 Access-Control-Request-Method 发送给服务器，另外浏览器会自行加上一个 Origin 请求地址。

服务端在接收到预检请求后，在响应头加入

- Access-Control-Allow-Origin：允许请求的源
- Access-Control-Allow-Methods：允许的请求方法
- Access-Control-Allow-Headers：允许的请求头

另外，服务端还可以通过`Access-Control-Max-Age`来设置一定时间内无须再进行预检请求，直接用之前的预检请求的协商结果即可。

浏览器再根据服务端返回的信息，进行决定是否再进行真实的请求。这个过程我们可以通过代理抓包软件或者浏览器的调试的网络中查看。

另外在HTTP响应头，凡是浏览器请求中携带了身份信息，而响应头中没有返回`Access-Control-Allow-Credentials: true`的，浏览器都会忽略此次响应。