# 理解HTTP协议中的Cookie及其与Session的区别

<https://itbilu.com/other/relate/Ny2IWC3N-.html>

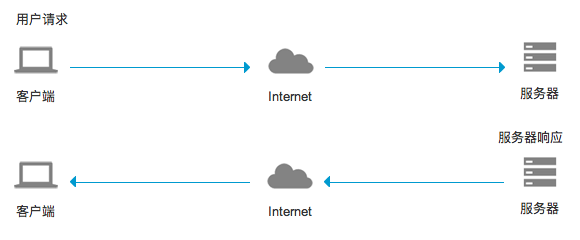
Cookie和Session都是为了解决HTTP协议的无状态问题，存储HTTP通讯中客户端与服务器之间的会话状态。不同的是Cookie依赖HTTP请求头与响应头实现且存储在客户端，而Session存储在服务器端，Session可以通过Cookie实现，也可以使用URL回写的方式实现。

1. [HTTP的无状态性](https://itbilu.com/other/relate/Ny2IWC3N-.html#problem)
   * [1.1 HTTP无状态带来的问题](https://itbilu.com/other/relate/Ny2IWC3N-.html#http-nostatus)
   * [1.2 无状态问题的解决](https://itbilu.com/other/relate/Ny2IWC3N-.html#cookie-session)
2. [Cookie](https://itbilu.com/other/relate/Ny2IWC3N-.html#cookie)
3. [Session](https://itbilu.com/other/relate/Ny2IWC3N-.html#session)

1. HTTP的无状态性

1.1 HTTP无状态带来的问题

HTTP协议是无状态的，客户端(浏览器)与服务器之间基于请求-响应实现数据交互，响应结束后两者之间的连接就会断开。由于每次会话都是一新连接，所以服务器无法从连接上获取会话状态。这样就会带来一些问题，如：页面发生跳转后服务器无从得知用户在上一个页面的输入、也无从得知用户是否登等。



1.2 无状态问题的解决

一个HTTP请求由三部分构成：*状态行*、[*请求头*](http://itbilu.com/other/relate/EJ3fKUwUx.html#http-request-headers)、*请求体*。其中*状态行*和*请求头*在所有的[HTTP请求方法](http://itbilu.com/other/relate/EkwKysXIl.html)中都存在，而*请求体*只有POST、PUT等部分请求方法中存在。

如，POST请求*IT笔录*网站时，服务器会收到如下结构的请求：

POST / HTTP/1.0 // 状态行

HOST: itbilu.com // 请求头

Accept-Encoding: gzip, deflate

user=myName&age=33 // 请求体

服务器收到客户端请求后，会对用户请求进行响应。一个HTTP响应也同样由三部分构成：*状态行*、[*响应头*](http://itbilu.com/other/relate/EJ3fKUwUx.html#http-response-headers)、*响应体*。与HTTP请求类似，*状态行*和*响应头*在所有HTTP请求响应中都会存在，而*响应体*只存在于部分HTTP请求方法的响应中。

如，*IT笔录*收到上面的请求后，会返回一个如下结构的HTTP响应：

HTTP/1.0 200 OK // 状态行

Content-type: text/html // 响应头

…… // 响应体

如上所示，HTTP请求/响应中，除正文外还有一个请求/响应头。而这个请求/响应头都是可设置的，根据这一特征我们可以*请求头*中加一个记录用户状态信息请求头，就可以实现用户状态的传递。

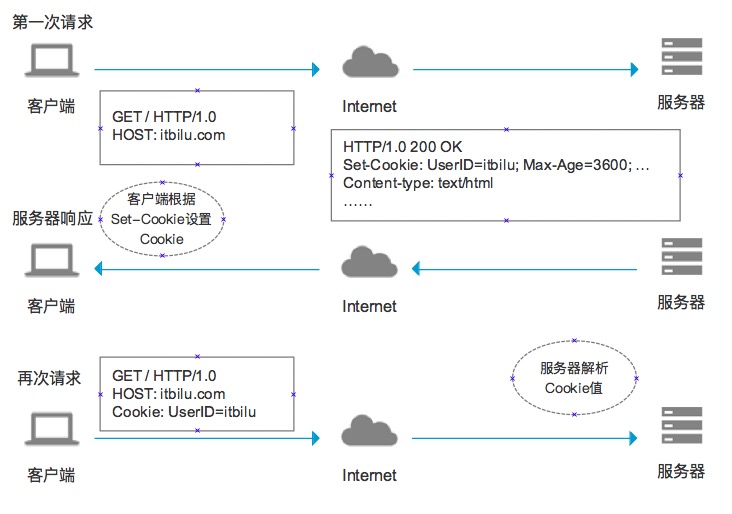
这也正是Cookie的实现方式。Cookie是存储在客户端的一个特殊的字符串，在发送HTTP请求时，这个字符串会添加到一个名为Cookie的请求头中一同发送到服务器。Session的实现方式也类似，不同的是状态信息存储在了服务器中，而只在客户端的Cookie中存储了一个表示该会话标识的SessionId。

2. Cookie

Cookie通过Cookie请求头和Set-Cookie响应头实现：

* Set-Cookie - 服务器响应头，用于告诉客户端要设置Cookie
* Cookie - 请求头，根据Set-Cookie设置并保存到客户端的Cookie值，会在再次发送HTTP请求时通过这个请求头一同发送到服务器

Cookie实现原理如下：



如上所示，用户请求客户端后，如果需要记录状态服务器会在请求响应中包含一个Set-Cookie响应头。客户端收到这个响应头到，会将Cookie保存到客户端。

对于一个需要保存状态的HTTP响应来说，请响应格式如下：

HTTP/1.0 200 OK

Set-Cookie: UserID=itbilu; Max-Age=3600; Version=1

Content-type: text/html

……

客户端收到这个响应头后，会将CookieUserID=itbilu进行存储，并设置超时时间为3600秒，而Version表示Cookie的版本。

再次请求服务器时，客户端会在请求头中包含一个Cookie请求头，其值就是之前从服务器返回的状态信息。

示例如下：

GET / HTTP/1.0

HOST: itbilu.com

Cookie: UserID=itbilu

更多关于Cookie的介绍，请参考：[Http Cookie机制及Cookie的实现原理](http://itbilu.com/other/relate/4J4n8fIPe.html" \t "_blank)。

3. Session

Cookie很好的解决了HTTP通讯中状态问题，但其本身也存在一些问题，如：

1. 客户端存储，可能会被修改或删除
2. 发送请求时，Cookie会被一起发送到服务器，当Cookie数据量较大时也会带来额外的请求数据量
3. 客户端对Cookie数量及大小有一定的限制

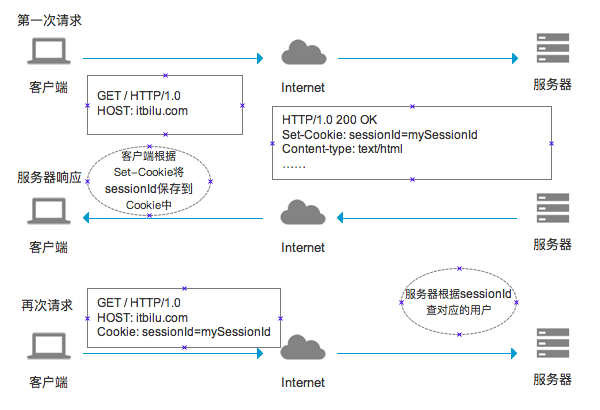
Session解决了Cookie的一些缺点。Session同样是为了记录用户状态，对于每个用户来说都会有相应的一个状态值保存在服务器中，而只在客户端记录一个sessionID用于区分是哪个用户的Session。

与Cookie相比Session有一定的优势：

1. Session值存储在服务器，相对来说更安全
2. 客户端发送给服务器的只有一sessionID，数据量更小

Session同样需要在客户端存储一个sessionID。可以这个值存储在Cookie，每次发送请求时通过Cookie请求头将其发送到服务器；也可以不使用Cookie，而将sessionID做为一个额外的请求参数，通过URL或请求体发送到服务器。

基于Cookie实现Session的实现原理如下：



由上可见，基于Cookie实现Session时，其本质上还是在客户端保存一个Cookie值。这个值就是sessionID，sessionID的名称也可按需要设置，为保存安全，其值也可能会在服务器端做加密处理。服务器在收到sessionID后，就可以对其解密及查找对应的用户信息等。