## [网络协议之Tcp、Http](https://www.cnblogs.com/ImBit/p/5513401.html)

**目录**

[**一、网络协议**](https://www.cnblogs.com/ImBit/p/5513401.html#one)

[**二、TCP（Transmission Control Protocol，传输控制协议）**](https://www.cnblogs.com/ImBit/p/5513401.html#two)

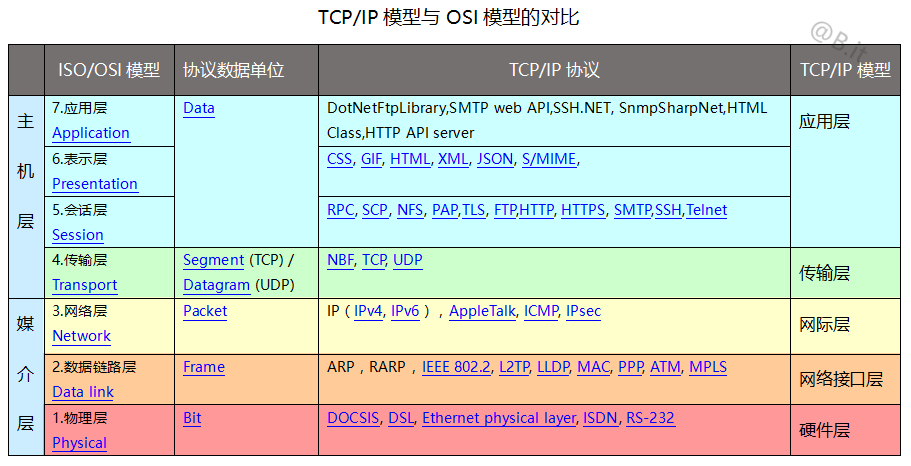
1. [TCP头格式](https://www.cnblogs.com/ImBit/p/5513401.html#two.one)
2. [TCP协议中的三次握手和四次挥手](https://www.cnblogs.com/ImBit/p/5513401.html#two.two)
3. [TCP报文抓取工具](https://www.cnblogs.com/ImBit/p/5513401.html#two.three)

[**三、HTTP（HyperText Transfer Protocol，超文本传输协议）**](https://www.cnblogs.com/ImBit/p/5513401.html#three)

1. [请求报文结构](https://www.cnblogs.com/ImBit/p/5513401.html#three.one)
2. [请求报文样例](https://www.cnblogs.com/ImBit/p/5513401.html#three.two)
3. [请求报文参数详解](https://www.cnblogs.com/ImBit/p/5513401.html#three.three)
4. [响应报文结构](https://www.cnblogs.com/ImBit/p/5513401.html#three.four)
5. [响应报文样例](https://www.cnblogs.com/ImBit/p/5513401.html#three.five)
6. [响应报文参数详解](https://www.cnblogs.com/ImBit/p/5513401.html#three.six)
7. [HTTP报文抓取工具](https://www.cnblogs.com/ImBit/p/5513401.html#three.seven)
8. [Session和Cookie](https://www.cnblogs.com/ImBit/p/5513401.html#three.eight)

[**四、相关资料**](https://www.cnblogs.com/ImBit/p/5513401.html#four)

**一、网络协议**

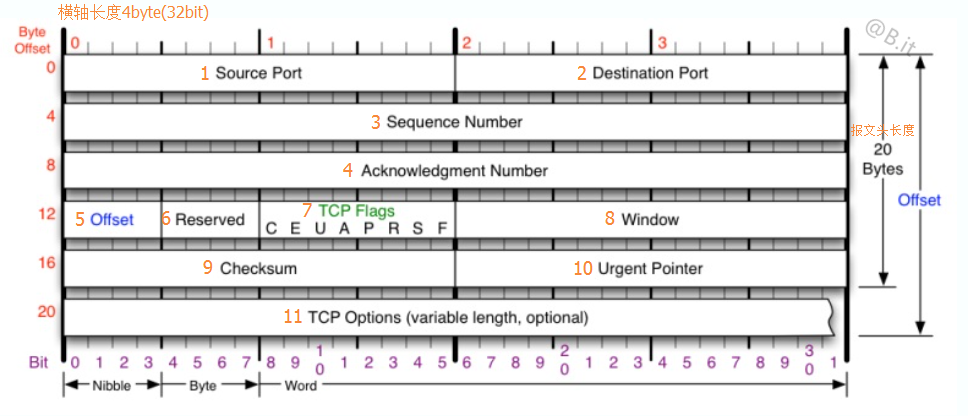
       国际标准化组织（International Standard Organization,ISO）公布了开放系统互连参考模型（OSI/RM）。OSI/RM是一种分层的体系结构，参考模型共有7层。  
TCP/IP(Transmission Control Protocol/Internet Protocol)作为Internet的核心协议。它是个协议族，包含多种协议。  
分层的基本想法是每一层都在它的下层提供的服务基础上提供更高级的增值服务，而最高层提供能运行分布式应用程序的服务。  
  
发送请求的过程是从最顶层（应用层）出发，每一层负责封装属于自己的信息到请求中，最后将一整个请求发送给对方。  
接收请求的过程是从最底层（网络接口层）开始，每一层的协议负责解析属于自己的东西，比如网际层（IP）处理ip信息，传输层（TCP）处理点对点的端口，应用层（HTTP）处理Request或Response的Line\Header\Body。

**二、TCP（Transmission Control Protocol，传输控制协议）**

      TCP是一种面向连接（连接导向）的、可靠的基于字节流的传输层通信协议。TCP将用户数据打包成报文段，它发送后启动一个定时器，另一端收到的数据进行确认、对失序的数据重新排序、丢弃重复数据。  
TCP的特点有：

* TCP是面向连接的运输层协议
* 每一条TCP连接只能有两个端点，每一条TCP连接只能是点对点的
* TCP提供可靠交付的服务
* TCP提供全双工通信。数据在两个方向上独立的进行传输。因此，连接的每一端必须保持每个方向上的传输数据序号。
* 面向字节流。面向字节流的含义：虽然应用程序和TCP交互是一次一个数据块，但TCP把应用程序交下来的数据仅仅是一连串的无结构的字节流。

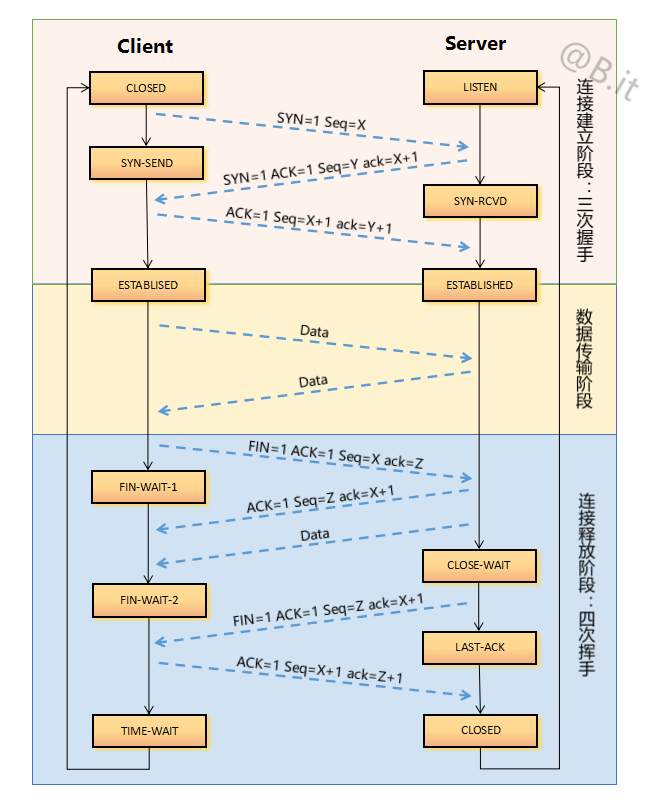
**1、TCP头格式**



（1）   Source Port（源端口号）：数据发起者的端口号，16bit。  
（2）   Destination Port（目的端口号）：数据接收者的端口号，16bit。  
（3）   Sequence Number（顺序号码，Seq）：用于在数据通信中解决网络包乱序（reordering）问题，以保证应用层接收到的数据不会因为网络上的传输问题而乱序（TCP会用这个顺序号码来拼接数据），32bit。  
（4）   Acknowledgment Number（确认号码，ack）：是数据接收方期望收到发送方在下一个报文段的顺序号码（Seq），因此确认号码应当是上次已成功收到顺序号码（Seq）加1，32bit。  
（5）   Offset（TCP报文头长度)：用于存储报文头中有多少个32bit(上图的一行)，存储长度为4bit，最大可表示（2^3+2^2+2^1+1）\*32bit=60bytes的报文头。最小取值5，5\*32bit=20bytes。  
（6）   Reserved（保留）：6bit, 均为0  
（7）   TCP Flags（TCP标志位）每个长度均为1bit  
          CWR：压缩，TCP Flags值0x80。  
          ECE：拥塞，0x40。  
          URG：紧急，0x20。当URG=1时，表示报文段中有紧急数据，应尽快传送。  
          ACK：确认，0x10。当ACK = 1时，代表这是一个确认的TCP包，取值0则不是确认包。  
          PSH：推送，0x08。当发送端PSH=1时，接收端尽快的交付给应用进程。  
          RST：复位，0x04。当RST=1时，表明TCP连接中出现严重差错，必须释放连接，再重新建立连接。  
          SYN：同步，0x02。在建立连接是用来同步序号。SYN=1， ACK=0表示一个连接请求报文段。SYN=1，ACK=1表示同意建立连接。  
          FIN：终止，0x01。当FIN=1时，表明此报文段的发送端的数据已经发送完毕，并要求释放传输连接。  
（8）   窗口：用来控制对方发送的数据量，通知发放已确定的发送窗口上限。  
（9）   检验和：该字段检验的范围包括头部和数据这两部分。由发端计算和存储，并由收端进行验证。  
（10） 紧急指针：紧急指针在URG=1时才有效，它指出本报文段中的紧急数据的字节数。  
（11） TCP选项：长度可变，最长可达40字节

备注：ISN（Inital Sequence Number）：初始化Sequence Number，发生在建立连接时。

**2、TCP协议中的三次握手和四次挥手**



特别注意

Seq：是发送方当前报文的顺序号码。  
ack：是发送方期望对方在下次返回报文中给回的Seq。

建立连接需要三次握手

第一次握手：客户端向服务端发送连接请求包，标志位SYN（同步序号）置为1，顺序号码为X=0。

第二次握手：服务端收到客户端发过来报文，由SYN=1知道客户端要求建立联机，则为这次连接分配资源。并向客户端发送一个SYN和ACK都置为1的TCP报文，设置初始顺序号码Y=0，将确认序号(ack)设置为上一次客户端发送过来的顺序号(Seq)加1，即X+1 = 0+1=1。

第三次握手：客户端收到服务端发来的包后检查确认号码(ack)是否正确，即第一次发送的Seq加1（X+1=1）。以及标志位ACK是否为1。若正确，服务端再次发送确认包，ACK标志位为1，SYN标志位为0。确认号码(ack)=Y+1=0+1=1，发送顺序号码(Seq)为X+1=1。Server收到后确认号码值与ACK=1则连接建立成功，可以传送数据了。

断开连接需要四次挥手

提醒：中断连接端可以是Client端，也可以是Server端。只要将下面两角色互换即可。  
第一次挥手：客户端给服务端发送FIN报文，用来关闭客户端到服务端的数据传送。将标志位FIN和ACK置为1，顺序号码为X=1，确认号码为Z=1。意思是说”我Client端没有数据要发给你了，但是如果你还有数据没有发送完成，则不必急着关闭Socket，可以继续发送数据。所以你先发送ACK过来。”

第二次挥手：服务端收到FIN后，发回一个ACK(标志位ACK=1),确认号码为收到的顺序号码加1，即X=X+1=2。顺序号码为收到的确认号码=Z。意思是说“你的FIN请求我收到了，但是我还没准备好，请继续你等我的消息" 这个时候客户端就进入FIN\_WAIT状态，继续等待服务端的FIN报文。

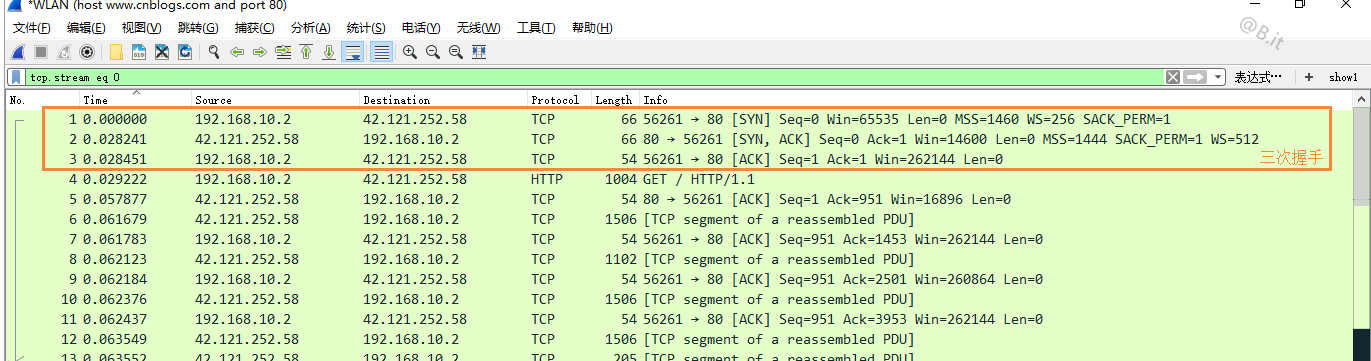
第三次挥手：当服务端确定数据已发送完成，则向客户端发送FIN报文，关闭与客户端的连接。标志位FIN和ACK置为1，顺序号码为Y=1，确认号码为X=2。意思是告诉Client端“好了，我这边数据发完了，准备好关闭连接了。”

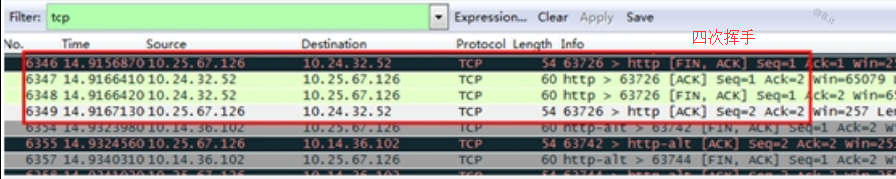
第四次挥手：客户端收到服务器发送的FIN之后，发回ACK确认(标志位ACK=1),确认号码为收到的顺序号码加1，即Y+1=2。顺序号码为收到的确认号码X=2。意思是“我Client端知道可以关闭连接了，但是我还是不相信网络，怕 Server端不知道要关闭，所以发送ACK后进入TIME\_WAIT状态，如果Server端没有收到ACK则可以重传。Client端等待了2MSL后依然没有收到回复，则证明Server端已正常关闭，那好，我Client端也可以关闭连接了。“（在TIME\_WAIT状态中，如果TCP client端最后一次发送的ACK丢失了，它将重新发送。TIME\_WAIT状态中所需要的时间是依赖于实现方法的。典型的值为30秒、1分钟和2分钟。等待之后连接正式关闭，并且所有的资源(包括端口号)都被释放。）

为什么关闭的时候却是四次挥（握）手？  
因为当Server端收到Client端的SYN连接请求报文后，可以直接发送SYN+ACK报文。其中ACK报文是用来应答的，SYN报文是用来同步的。但是关闭连接时，当Server端收到FIN报文时，很可能并不会立即关闭SOCKET，所以只能先回复一个ACK报文，告诉Client端，"你发的FIN报文我收到了"。只有等到我Server端所有的报文都发送完了，我才能发送FIN报文，因此不能一起发送。故需要四步握手。

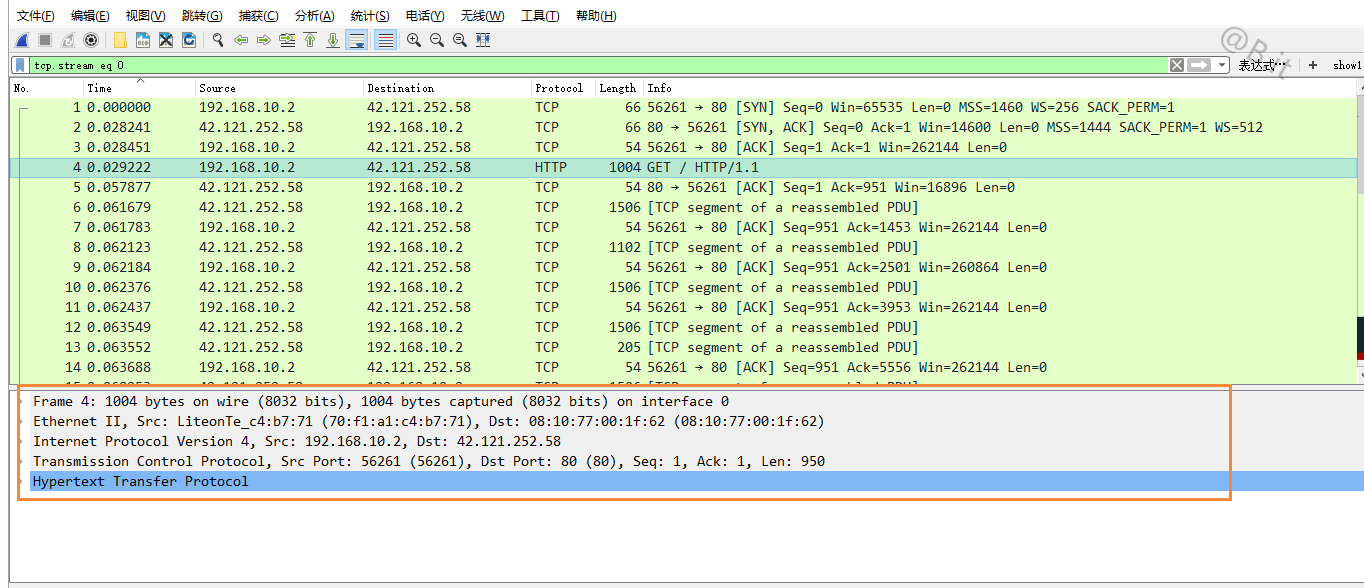
**3、TCP报文抓取工具：Wireshark**

捕获过滤器中填入表达式：host www.cnblogs.com and port 80（80等效于http）  
有多个TCP流时在显示过滤器中填入表达式：tcp.stream eq 0 筛选出第一个TCP流（包含完整的一次TCP连接：三次握手和四次挥手）





每条记录都有如下协议层  
（1） Frame:   物理层的数据帧概况  
（2）Ethernet II: 数据链路层以太网帧头部信息  
（3） Internet Protocol Version 4: 互联网层IP包头部信息  
（4）Transmission Control Protocol:  传输层的数据段头部信息，此处是TCP  
（5） Hypertext Transfer Protocol:  应用层的信息，此处是HTTP协议



**三、HTTP（HyperText Transfer Protocol，超文本传输协议）**

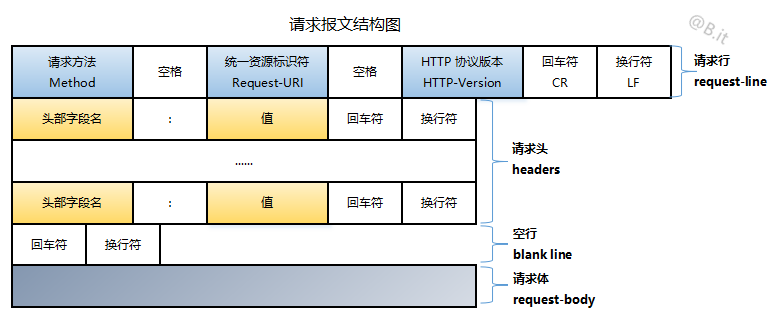
      HTTP是一个应用层协议，虽然在2015年已推出HTTP/2版本，并被主要的web浏览器和web服务器支持。但目前使用最广泛的还是HTTP/1.1版本。有关历史请查阅[这里](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol#History)。  
它的主要特点可概括如下：

* 支持客户/服务器模式。
* 简单快速：客户向服务器请求服务时，只需传送请求方法和路径。由于HTTP协议简单，使得HTTP服务器的程序规模小，因而通信速度很快。
* 灵活：HTTP允许传输任意类型的数据对象。正在传输的类型由Content-Type加以标记。
* 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。
* 无状态：HTTP协议是无状态协议。无状态是指协议对于事务处理没有记忆能力。缺少状态意味着如果后续处理需要前面的信息，则它必须重传，这样可能导致每次连接传送的数据量增大。另一方面，在服务器不需要先前信息时它的应答就较快。为了解决这个问题， Web程序引入了Cookie机制来维护状态。

另外，HTTP请求报文和响应报文都是由开始行（对于请求消息，开始行就是请求行，对于响应消息，开始行就是状态行），消息报头（可选），空行（只有CRLF的行），消息正文（可选）组成。将在下面详细讲解。

**1、请求报文结构**

报文中的数据都使用ASCII编码，各个字段的长度是不确定的（除了作为结尾的CRLF外，不允许出现单独的CR或LF字符）。



**2、请求报文样例**

[复制代码](javascript:void(0);)

POST /search HTTP/1.1

Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-excel, application/vnd.ms-powerpoint,

application/msword, application/x-silverlight, application/x-shockwave-flash, \*/\*

Referer: http://www.google.cn/

Accept-Language: zh-cn

Accept-Encoding: gzip, deflate

User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 2.0.50727; TheWorld)

Host: www.google.cn

Connection: Keep-Alive

Cookie: PREF=ID=80a06da87be9ae3c:U=f7167333e2c3b714:NW=1:TM=1261551909:LM=1261551917:S=ybYcq2wpfefs4V9g;

NID=31=ojj8d-IygaEtSxLgaJmqSjVhCspkviJrB6omjamNrSm8lZhKy\_yMfO2M4QMRKcH1g0iQv9u-2hfBW7bUFwVh7pGaRUb0RnHcJU37y-

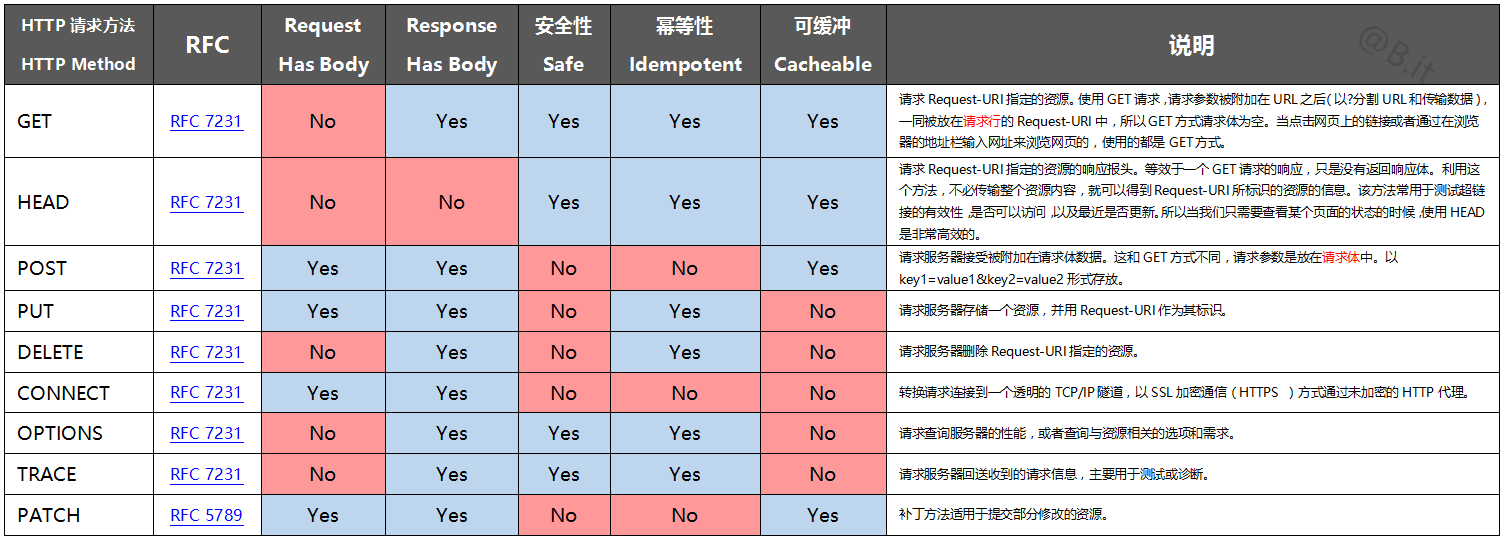
FxlRugatx63JLv7CWMD6UB\_O\_r

hl=zh-CN&source=hp&q=domety

[复制代码](javascript:void(0);)

**3、请求报文参数详解**

请求方法

所有请求方法名称全为大写，目前有9种：  
  
备注  
安全性：https://en.wikipedia.org/wiki/Hypertext\_Transfer\_Protocol  
幂等性：表示的操作至多只会被处理一次，每次调用都将返回第一次调用时的处理结果。  
关于HTTP请求GET和POST的区别  
(1).提交形式：  
   GET提交的数据会放在URL之后，以?分割URL和传输数据，参数之间以&相连，如EditPosts.aspx?name=test1&id=123456.  POST方法是把提交的数据放在HTTP包的Body中.  
(2).传输数据的大小：  
   HTTP协议本身没有对传输的数据大小进行限制，HTTP协议规范也没有对URL长度进行限制。 而在实际开发中存在的限制主要有：  
   GET:特定浏览器和服务器对URL长度有限制，例如IE对URL长度的限制是2083字节(2K+35)。对于其他浏览器，如Netscape、FireFox等，理论上没有长度限制，其限制取决于操作系统的支持。  
   因此对于GET提交时，传输数据就会受到URL长度的限制。  
   POST:由于不是通过URL传值，理论上数据不受限。但实际各个WEB服务器会规定对post提交数据大小进行限制，Apache、IIS6都有各自的配置。  
(3).安全性：  
    POST的安全性要比GET的安全性高，具有真正的Security的含义。而且通过GET提交数据，用户名和密码将明文出现在URL上，因为登录页面有可能被浏览器缓存，其他用户浏览历史纪录就可以拿到账号和密码了。

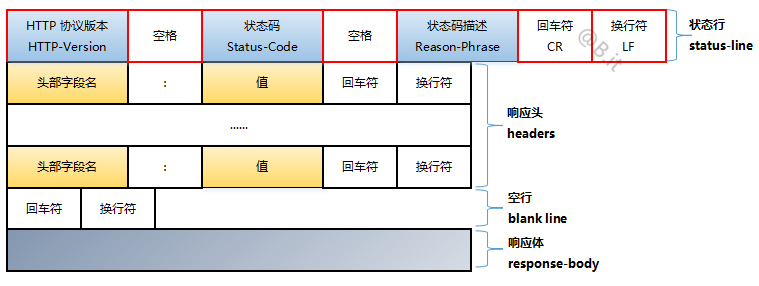
请求报头域

报头域指头部中的Key，且不分大小写。



**4、响应报文结构**

如所见，响应报文结构与请求报文结构唯一真正的区别在于第一行中用状态信息代替了请求信息。状态行（status line）通过提供一个状态码来说明所请求的资源情况。



**5、响应报文样例**

[复制代码](javascript:void(0);)

HTTP/1.1 200 OK

Date: Mon, 23 May 2005 22:38:34 GMT

Content-Type: text/html; charset=UTF-8

Content-Encoding: UTF-8

Content-Length: 138

Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT

Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)

ETag: "3f80f-1b6-3e1cb03b"

Accept-Ranges: bytes

Connection: close

<html>

<head>

<title>An Example Page</title>

</head>

<body>

Hello World, this is a very simple HTML document.

</body>

</html>

[复制代码](javascript:void(0);)

**6、响应报文参数详解**

响应状态码

状态代码由三位数字组成，第一个数字定义了响应的类别，且有五种可能取值。  
1xx：指示信息--表示请求已接收，继续处理。  
2xx：成功--表示请求已被成功接收、理解、接受。  
3xx：重定向--要完成请求必须进行更进一步的操作。  
4xx：客户端错误--请求有语法错误或请求无法实现。  
5xx：服务器端错误--服务器未能实现合法的请求。  
常用状态码：

200 OK：成功返回状态，对应，GET,PUT,PATCH,DELETE。  
201 created  - 成功创建。  
302 Found：重定向，新的URL会在response中的Location中返回，浏览器将会使用新的URL发出新的Request。

                 例如在IE中输入http://www.google.com. HTTP服务器会返回304， IE取到Response中Location header的新URL, 又重新发送了一 个 Request.  
304 Not Modified：代表上次的文档已经被缓存了， 还可以继续使用。  
400 bad request   - 请求格式错误。  
401 unauthorized   - 未授权。  
403 forbidden   - 鉴权成功，但是该用户没有权限。  
404 not found - 请求的资源不存在。  
405 method not allowed - 该http方法不被允许。  
410 gone - 这个url对应的资源现在不可用。  
415 unsupported media type - 请求类型错误。  
422 unprocessable entity - 校验错误时用。  
429 too many request - 请求过多。  
500 Internal Server Error：服务器发生了不可预期的错误。  
503 Server Unavailable：服务器当前不能处理客户端的请求，一段时间后可能恢复正常。

其它状态码请查阅：<https://en.wikipedia.org/wiki/List_of_HTTP_status_codes>

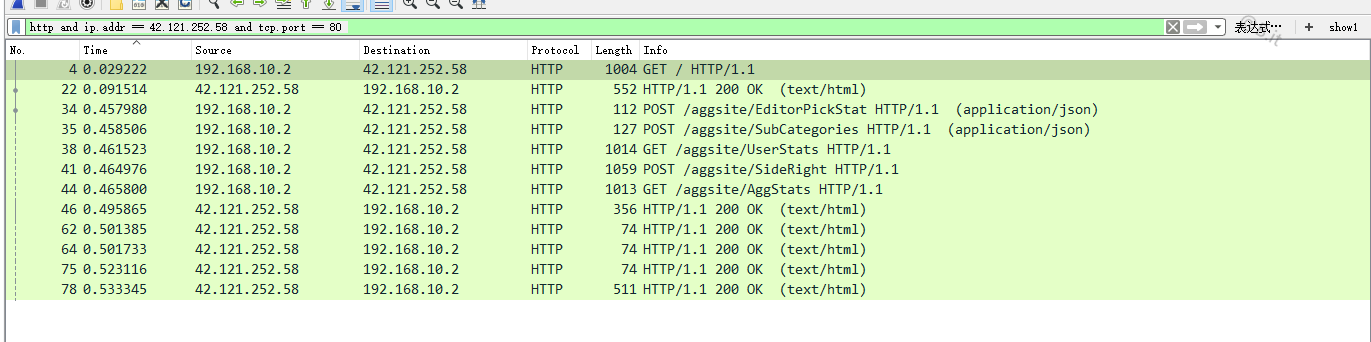
响应报头域

报头域指头部中的Key，且不分大小写。



**7、HTTP报文抓取工具**

Wireshark、Fiddler、HttpWatch（需结合IE）、Telnet  
Wireshark：  
在显示过滤器中填入表达式：http and ip.addr == 42.121.252.58 and tcp.port == 80 过滤出http的响应和请求流程



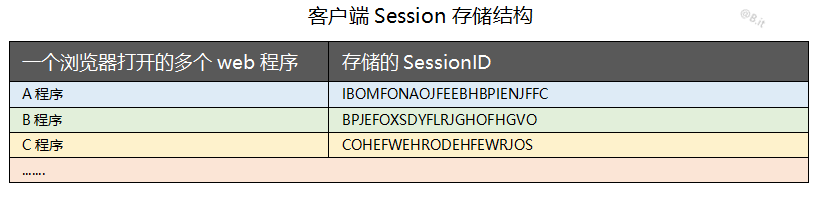
**8、Session和Cookie**

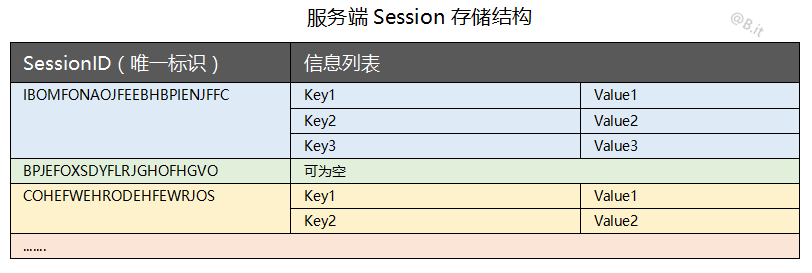
说到HTTP，就不得不提Session和Cookie。但严格来说，Session和Cookie并不是http协议的一部分。由于HTTP协议设计原则是无状态的，但是近年来出现了种种需求，其中cookie的作用就是为了解决HTTP协议无状态的缺陷所作出的努力。后来出现的session机制则是又一种在客户端与服务器之间保持状态的解决方案。 具体来说cookie机制采用的是在客户端保持状态的方案，而session机制采用的是在服务器端保持状态的方案。同时我们也看到，由于采用服务器端保持状态的方案在客户端也需要保存一个标识，所以session机制可能需要借助于cookie机制来达到保存标识的目的，但实际上它还有其他选择。

Session

Session是可以存储针对于某一个用户的浏览器以及通过其当前窗口打开的任何窗口具有针对性的用户信息存储机制。   
通常大家认为，只要关闭浏览器，session就消失，其实这是错误的理解。对session来说也是一样的，除非程序通知服务器删除一个session，否则服务器会一直保留。由于关闭浏览器不会导致session被删除，迫使服务器为seesion设置了一个失效时间，当距离客户端上一次使用session的时间超过这个失效时间时，服务器就可以认为客户端已经停止了活动，才会把session删除以节省存储空间.

(1)第一次访问某个web站点资源时，客户端提交没有带SessionID的请求（请求报文头没有Cookie头域信息）。  
  而web服务器会检查是否有SessionID过来，没有则创建SessionID，并根据web程序自身定义在请求哪个资源时添加属于当前会话的信息（也可为空），这个信息列表以SessionID作为标识。然后将SessionID返回给客户端（通过响应报文头的Set-Cookie头域）。  
(2 )客户端再次访问同个web站点时，提交带有SessionID的请求（通过Cookie头域存储SessionID）。由服务端判断session是否失效，如果未失效，可查询属于当前会话的信息列表。如果失效，则创建新的session（产生新的SessionID），而原先的session（包含session带的信息列表）则丢失，无法访问。





Cookie

保存SessionID的方式可以采用Cookie，这样在交互过程中浏览器可以自动的按照规则把这个SessionID发回给服务器。Cookie的命名方式类似于SessionID。有时Cookie被人为的禁止，所以出现了其他机制以便在Cookie被禁止时仍然能够把SessionID传递回服务器。这种技术叫做URL重写，就是把SessionID直接附加在URL路径的后面，附加方式也有两种，一种是作为URL路径的附加信息，表现形式为http://www.wantsoft.com/index.asp;jsessionid= ByOK3vjFD75aPnrF7C2HmdnV6QZcEbzWoWiBYEnLerjQ99zWpBng!-145788764 。  
另一种是作为查询字符串附加在URL后面，表现形式为http://www.wantsoft.com/index?js ... 99zWpBng!-145788764 。

**四、相关资料**

* 《系统架构设计师教程》
* 《C#网络应用编程》（第2版）
* [Hypertext Transfer Protocol -- HTTP/1.1](http://www.w3.org/Protocols/rfc2616/rfc2616.html)
* [OSI model](https://en.wikipedia.org/wiki/OSI_model)
* [TCP/IP Reference](https://nmap.org/book/tcpip-ref.html)
* [wireshark抓包图解 TCP三次握手/四次挥手详解](http://www.seanyxie.com/wireshark%E6%8A%93%E5%8C%85%E5%9B%BE%E8%A7%A3-tcp%E4%B8%89%E6%AC%A1%E6%8F%A1%E6%89%8B%E5%9B%9B%E6%AC%A1%E6%8C%A5%E6%89%8B%E8%AF%A6%E8%A7%A3/)
* [TCP 的那些事儿（上）](http://kb.cnblogs.com/page/209100/)
* [TCP协议中的三次握手和四次挥手(图解)](http://blog.csdn.net/whuslei/article/details/6667471)
* [Hypertext Transfer Protocol](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol)
* [HTTP 协议详解](http://kb.cnblogs.com/page/130970/)
* [HTTP请求报文和HTTP响应报文](http://blog.csdn.net/zhangliang_571/article/details/23508953)
* [HTTP协议详解（经典）](http://blog.csdn.net/gueter/archive/2007/03/08/1524447.aspx)
* [HTTP协议之Session和Cookie](http://leowzy.iteye.com/blog/1216841)