# HTTP 图解

# 原文链接：[blog.csdn.net](https://link.juejin.im/?target=http://blog.csdn.net/u014346301/article/details/53939872" \t "https://juejin.im/entry/_blank)

# 本节内容

俗话说好的开发，底层知识必须过硬，不然再创新的技术，你也理解不深入，比如**[Python](https://link.juejin.im/?target=http://lib.csdn.net/base/python" \o "Python知识库" \t "https://juejin.im/entry/_blank)** web开发工程师，想要学习任何一个框架，底层都是http和socket，底层抓牢了，学起来会很轻松，所以楼主今天特意写了一篇底层的博客，来源于《HTTP图解》这本书。下载内容在本节末尾，可以自行前去下载观看，比我个人了解得多要全面很多，不过不要忘记点赞哈！！

本文只要内容：

* 了解web及网络基础
* 简单的http协议

# 了解web及网络基础

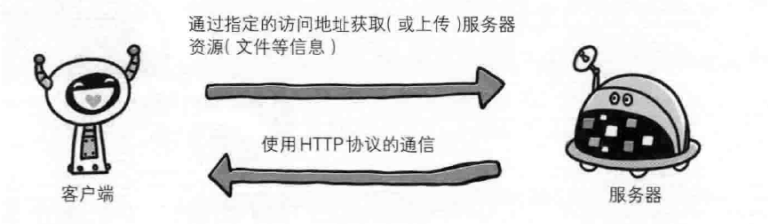
### 1、使用http协议访问web

当我们用网页浏览器（web browser）的地址栏中输入URL时，Web页面的呈现过程：

* 输入URL时，可以看到web页面，即使不了解运作原理，也能看到页面
* 输入URL后，信息会被送往某处
* 然后从某处获得的回复，内容就会显示在web页面上

web页面不可能凭空显示出来，根据web浏览器地址栏中指定的URL，web浏览器会从服务器端获取文件资源等信息，从而显示web页面。

像这一套流程就可以称作是客户端（client）向服务端（server）获取服务器资源



web使用一种名为HTTP（HyperText Transfer Protocol），超文本传输协议作为规范，完成从客户端到服务端等一系列运作过程。而协议是指规则的约定。可以理解成，web是建立在HTTP协议上通信的。

### 2、网络基础TCP/IP

为了了解HTTP，必须的了解TCP/IP协议族。

通常使用的网络实在TCP/IP协议族的基础上运作的。而HTTP就属于他的一个子集。

#### 2.1、TCP/IP 协议族

计算机与网络设备要相互通信，双方就必须基于相同的方法。比如：如何探测到通信目标、由哪一方先发起通信、使用哪种语言等等好多规则。然而这种规则就是所谓的协议（protocol），你要用我就按照我协议的规则来。常用协议有：tcp，http，ftp，dhcp，dns，ip，pppoe，udp，snmp，icmp等等。

像这样把与互联网关联的协议集合起来总称TCP/IP，TCP/IP实在IP协议通信过程中，使用到的协议族的统称

#### 2.2、TCP/ip的分层管理

TCP/IP协议族重要的一点就是分层。TCP/IP协议族按层次划分分别是：应用层、传输层、网络层和数据链路层。把塔层次化是有好处的。比如：当某个地方需求变更时，就必须吧所有的整体替换掉。而分层之后只需要把变动的层替换掉即可。下面介绍一下每个层次的作用

****应用层****

应用层决定了向用户提供应用服务时通信的活动。比如：ftp，dns服务就是其中的两类。HTTP协议也处于该层。

****传输层****

传输层对上层应用层，提供处于网络连接中的两台计算机之间的数据传输。在传输层有两个性质不同的协议：TCP和UDP。

****网络层****

网络层用来处理网络上流动的数据包。数据包是网络传输的最小单位。该层规定了通过怎样的路径到达对方计算机，并把数据包发送给对方。

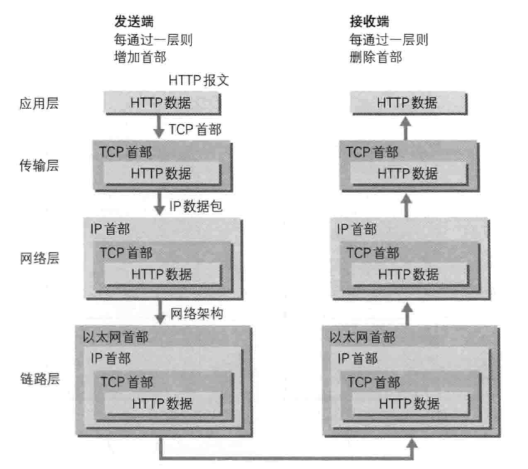
****链路层****

用来处理链接网络的硬件部分。包括**[操作系统](https://link.juejin.im/?target=http://lib.csdn.net/base/operatingsystem" \o "操作系统知识库" \t "https://juejin.im/entry/_blank)**、硬件的设备驱动、网卡及光纤等物理可见部分。

#### 2.3、TCP/IP通信传输流

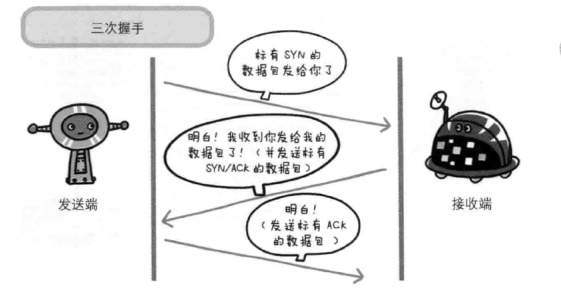
比如说我想看百度网页：流程如下：

* 客户端在应用层发出一个想看某个WEB页面的HTTP请求的时候；
* 接着为了传输方便，在传输层把应用层收到的数据进行分割，并在各个报文上打上标记序号及端口号后转发给网络层；
* 在网络层增加作为通信目的的MAC地址后转发给链路层，这样一来发送网络的通信请求就准备齐全了；
* 接收端的服务器在链路层接收到数据，按序往上层发送，一直到应用层；
* 当传输到应用层，才能算真正接收到由客户端发送过来的HTTP请求了。



发送端在层与层之间传输数据时，每经过一层时必定会打上一个该层的首部信息。接收端没经过一层，会把消去。这种数据信心包装叫做封装。

TCP/IP传输是三次握手，四次断开。一张图搞定一切，哈哈



### 3、URI和URL

URI用字符串标示某一互联网资源，而URL表示资源的地点。可见URL是URI的子集。

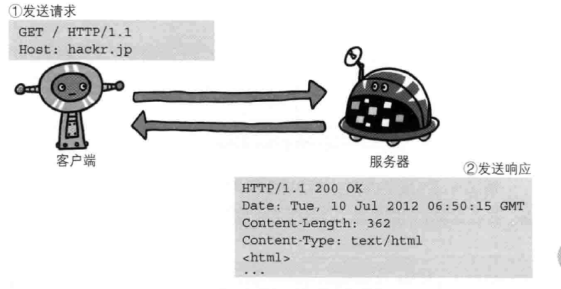
URI要使用涵盖全部必要信息的URI、绝对URL以及相对URL。相对URL是指从浏览器中基本URI处理的URL，来先看下URI的格式



# 简单的HTTP协议

### 1、通信过程

HTTP协议规定，请求从客户端发出，最后服务器端相应该请求并返回。换个意思就是客户端建立通信的，服务器端在没有接收到请求之前不会发送响应。来一个示例：



下面的内容是客户端想服务端发的求求内容

GET / HTTP/1.1

HOST: HACKR.JP

起始行的get表示请求访问服务器的类型，成为方法。随后的字符串指明了请求访问的资源对象，也叫请求URL，后面是版本号，用来提示客户端使用的HTTP协议功能

总结：请求报文是由请求方法、请求URL、协议版本、可选的请求头和内容实体构成的。

接下来服务器端接收到请求，会将请求内容的处理结果以相应的形式返回。

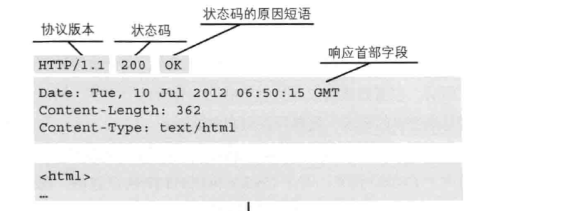
HTTP /1.1 200 OK

Date：Tue，10 JUL 2016 10:50:20 GMT

Content-length：398

Content-Type：text/html

来用图片详细讲解一下：



### 2、HTTP是不保存状态的协议

HTTP是一种不报错状态，即无状态协议。http协议自身不对请求和相应之间的通信状态进行保存，也就是说在HTTP这个级别，协议对于发送过得请求和响应不做持久化处理。

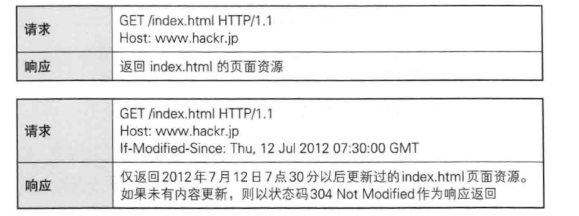
使用HTTP协议，每当有新的请求发送时，就会有对应的新响应产生。协议本身是并不保留之前的请求或响应报文的信息。这是为了处理大量事务，确保协议的可伸缩性，而特意把HTTP协议设计成如此简单的。但是也存在弊端，当业务处理变得棘手的情况多了,比如用户登录一家网站，即使他跳转到别的页面后，也需要保持登录状态，因此就有了cookie的诞生。有了cookie再用http协议通信，就可以管理状态了。

### 3、HTTP方法

#### 3.1、GET获取资源

get方法是用来请求访问已被URL识别的资源。指定的资源经服务端解析后返回响应内容。也就是说，请求的资源是文本，那就保持原样返回；如果像cgi那样的程序，则返回经过执行后的输出结果

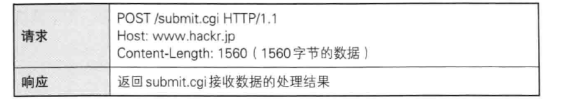
使用get方法的请求响应的例子：



#### 3.2、POST传输实体主体

post方法用来传输实体的主体。虽然用get方法也可以传输实体的主体，但是一般不用get方法进行传输，而是用post方法。虽说post的功能与get很相似，但是post的主要目的并不是获取相应的主体内容。

使用post方法请求的例子：



#### 3.3、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，要求在请求报文的主体中包含文件内容，然后保存到请求url指定的位置。由于put的方法不带验证机制，任何人都可以上传文件，存在安全性问题，因此一般的web网站不适用该方法。

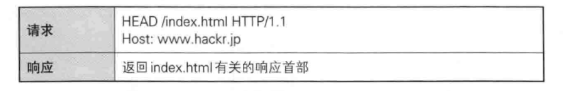
使用put方法请求的例子：



#### 3.4、head获取报文首部

head和get方法一样，只是不返回报文主体部分，用于确认url的有效性及资源更新的日期时间等。

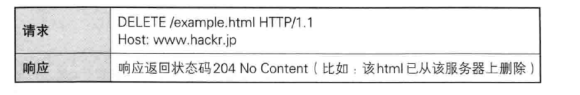
使用head方法请求的例子：



#### 3.5、DELETE删除文件

delete方法用来删除文件，是与put相反的方法，delete方法按照请求url删除指定的资源。其本质和put方法一样不带验证机制，所以不适用delete方法。

使用delete方法请求的例子：

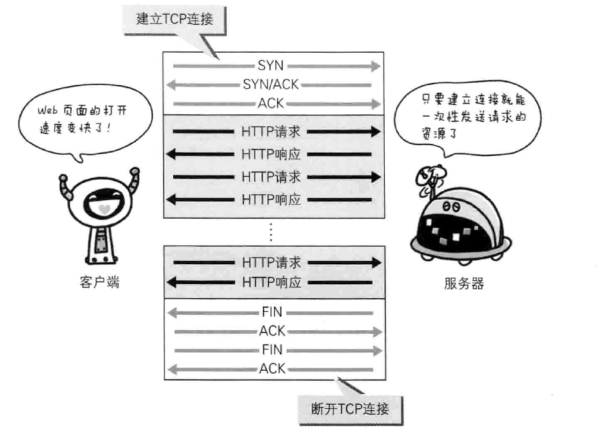


下面列举一些方法：其中link和unlink已被HTTP/1.1废弃 ，不在支持



### 4、持久化

当频繁访问web网页的时候，每次都要进行TCP/IP通信，tcp/ip通信三次握手四次断开，代价是很昂贵的，增加了通信量的开销，为解决上述TCP连接的问题，HTTP就诞生了持久连接的方法。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

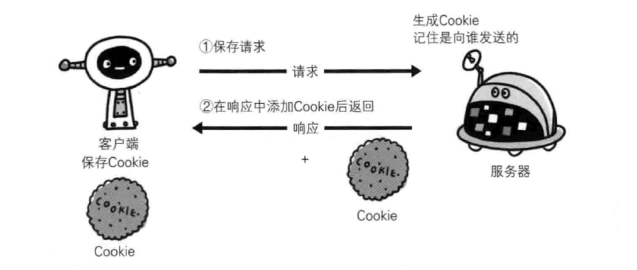


持久化连接减少了TCP连接的重复建立和断开所造成的额外开销，减轻了服务端的负载。另外，减少开销的那部分时间，使HTTP请求和响应能够更早的结束，这样web页面的显示速度也就相应的提高了。

### 5、使用cookie的状态管理

cookie技术通过在请求和相应报文中写入cookie信息来控制客户端的状态。cookie会根据从服务端发送的相应报文内的一个叫做set-cookie的首部字段信息，通知客户端保存cookie。当下次客户端再往服务器发送请求的时候，客户端会自动在请求头加入cookie值后发送出去。下面带图分析下过程

* 没有cookie状态下的请求



* 第二次以后（存有cookie信息状态）的请求

