**HTTP协议介绍**

# 1、HTTP概述及网络基础

HTTP协议从最初期的HTTP/0.9，发展到现在HTTP/1.1版本，目前的最新版本为RFC2616版本。

## 1.1 网络基础TCP/IP

通常使用的网络（包括互联网）是在TCP/IP协议族基础上运作的。而HTTP属于它内部的一个子集。

计算机与网络设备之间要通信，双方就必须要按照某种既定规则进行运行。例如：如何探测到通信目标、由哪一边先发起通信、使用哪种语言、如何结束通信等等都需要进行规定。而这种规则我们就叫做协议。

## 1.2 TCP/IP的分层管理

TCP/IP协议族中最重要的一点就是分层。包括应用层、传输层、网络层和数据链路层。

分层的设计好处就是，某个地方需要修改时，只需要把相应的层进行修改即可；同时分层使得设计也变得简单，每个层只需要清楚自己所处理的任务即可，而不需要考虑其他层次是怎样工作和工作的质量。

* **应用层**

应用层决定了向用户提供应用服务时的通信活动。包括FTP协议、DNS协议、HTTP协议。

* **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。包括TCP和UDP协议。

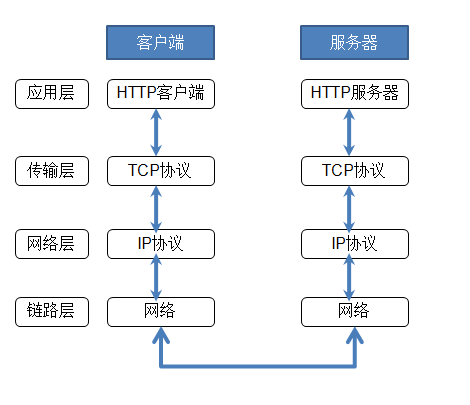
* **网络层**

网络层用来处理网络的数据包，该层规定了数据包应该通过怎样的路径到达计算机，并把数据包传递给对方。

* **数据链路层**

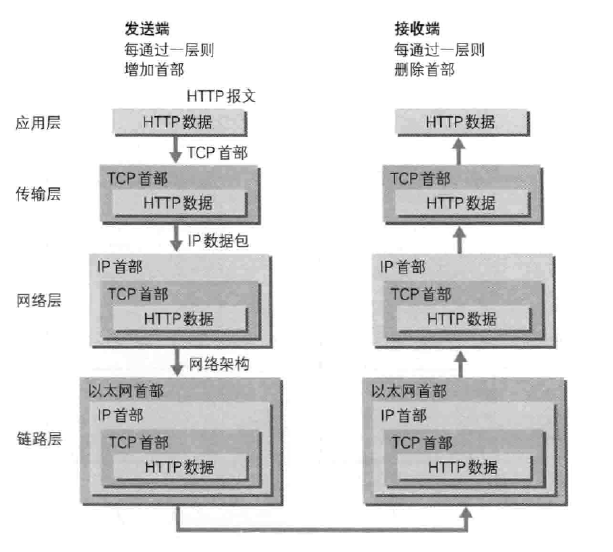
数据链路层用来处理连接网络的硬件部分。包括控制操作系统、硬件的设备驱动及光纤等可见部分。硬件上的范畴均属数据链路层的作用范围之内。

## 1.3 TCP/IP通信传输流



利用TCP/IP协议族进行网络通信时。发送端从应用层网下走，接收端则从数据链路层往上走。

下面以HTTP协议举例来说明访问web页面的HTTP请求。



## 1.4 其他协议介绍

这里介绍几个与HTTP密切相关的几个协议，包括：IP、TCP和DNS协议。

### 1.4.1 负责传输的IP协议---IP网际协议，工作在网络层

IP协议的作用是把各种数据包传输给对方。而为了保证能正确传递给对方，就需要IP地址和MAC地址。

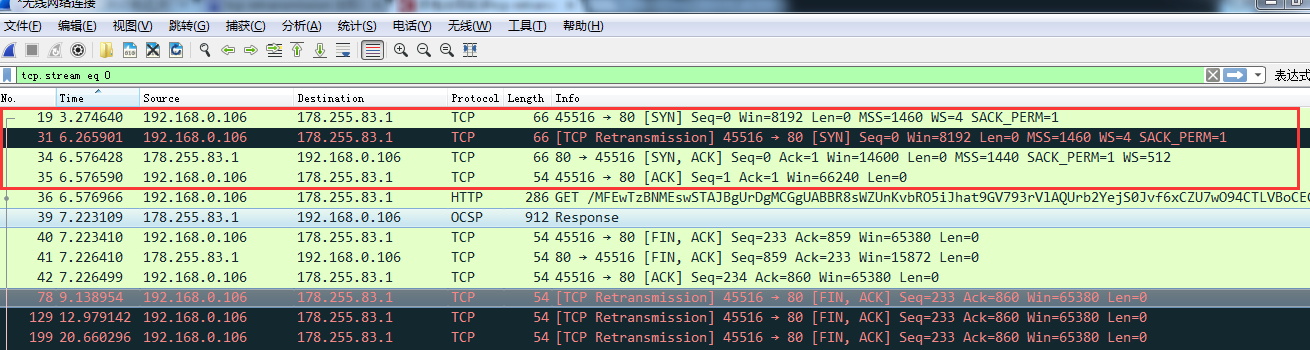
IP地址指明了节点被分配到的地址，而MAC地址是网卡的固定地址。IP地址可变，但是MAC地址不会变。

而计算机之间的通信还需要通过ARP（地址解析协议），通过ARP协议可以查找出对应IP地址的MAC地址。

### 1.4.2 确保可靠性的TCP协议---TCP协议，工作在传输层

TCP工作在传输层，提供可靠的字节流服务。所谓字节流服务，是数据包在网络传输过程中，会将一个大包进行切块，切成报文段为单位的数据包进行管理。TCP协议为了更容易的传输大数据才将数据包分割，而且TCP协议能够确认数据是否最终送达到对方。

为了准确无误的将数据送达目标处，TCP协议采用3次握手策略。用TCP协议把数据包发送出去后，会向对方确认是否成功送达。握手过程中使用TCP的标识------SYN和ACK。发送端首先发送一个带SYN标识的数据包给对方。接收端收到后，回传一个带有SYN/ACK标识的数据包以传达确认信息。最后发送端再回传一个带ACK标识的数据包，代表握手结束。若握手的过程中通信中断，TCP协议会再次以相同的顺序发送相同的数据包。



上图中红线框部分就是一个完整的TCP3次握手，中间的"tcp retransmission"因为网络状况不好而导致的重发包。

### 1.4.3 负责域名解析------DNS协议，工作在应用层

DNS协议与HTTP协议一样工作在应用层，提供域名到IP地址的解析服务。通过域名或IP地址都可以访问web页面，但是相对而言人记住一个域名比记住一串数字更容易。而计算机更擅长处理数字，因此DNS协议才诞生了。DNS协议可以通过域名查找到IP地址，也可以通过IP地址查找到域名。

### 1.4.4 完整的数据通信过程

在我们访问一个web页面时：

**客户端**

首先，通过DNS协议，获取到目标地址的IP地址；

其次，通过http协议生成将要访问的页面的数据包；

第三，TCP协议将请求的数据包分割成多个较小的报文；

第四，IP协议搜索对方地址，一边中转一边传递数据；（路由器--路由选择）。

**服务器端**

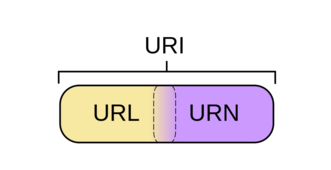
第五，TCP协议将受到的报文进行重组，组合成原始的数据包；

第六，HTTP协议读取数据包的内容，服务器处理该请求后，进过上述同样的路径将客户端所想要的结果传递给对方，这样就完成了一次完整的通信过程。

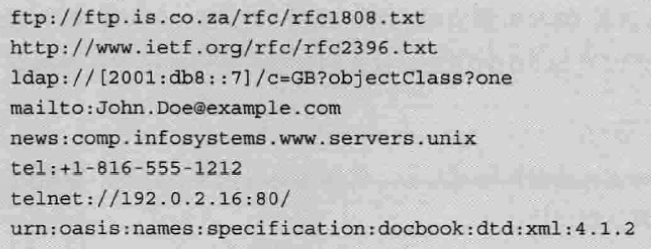
## 1.5 URL和URI的关系

URL中文名：统一资源定位符；URL中文名：统一资源标识符。

访问web页面时，在地址栏中输入的一串(http://www.baidu.com)就是URL。一般不怎么见这个URI，都是熟悉URL。



从上图中可以看到URL是URI的子集，即所有的URL都是URI，但URI不一定就是URL。URI就是由某个协议方案表示的资源定位标识符。协议方案是指访问资源所使用的协议的名称。如http、ftp、telnet、file等。URI用字符串表示某一互联网资源，而URL表示资源的地点。如下图所列举的URI：



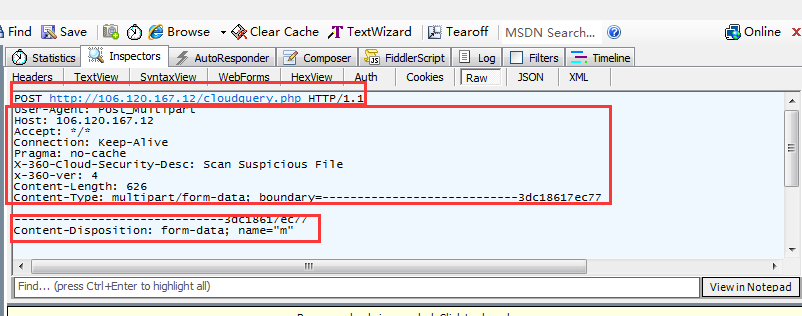
按照通常的理解，可以把URI当成URL来理解。

# 2、HTTP协议简单介绍

HTTP协议规定，请求从客户端发出，最后服务端响应该请求并返回。换句话说肯定是先有客户端发出请求，建立通信连接，然后才会有服务端的响应内容。

## 2.1 HTTP请求和响应分析

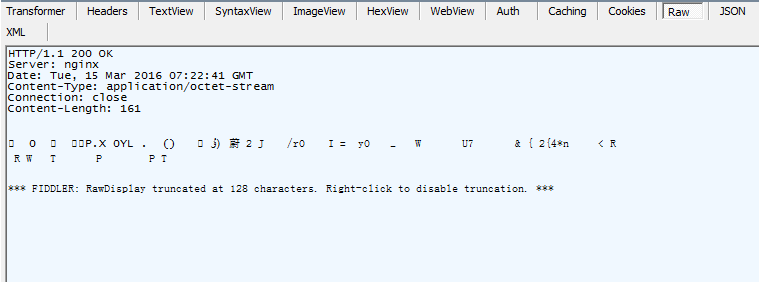
下面来看个具体的示例：



POST http://106.120.167.12/cloudquery.php HTTP/1.1 表示：采用POST方法，通过HTTP协议，访问http://106.120.167.12/下的cloudquery.php文件。

可看出请求报文是由请求方法、请求URL、协议版本、可选的请求首部字段和内容实体构成。

接下来看下对应的响应请求内容：



可看到响应请求内容是由协议版本、状态码、用以解释状态码的原因短语、可选的响应首部字段以及实体主体构成。

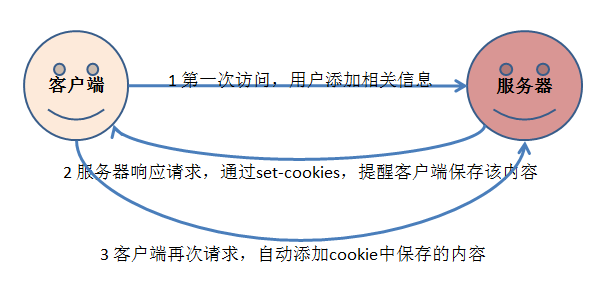
## 2.2 HTTP协议是不保存状态的协议

HTTP协议是一种不保存状态的的协议。协议自身不对请求和响应的通信状态进行保存，也就是说协议对于发送的请求或响应都不做持久化处理。

随着web技术的不断发展，因无状态而导致业务处理变得棘手起来，试想一下，你登录淘宝网，每次打开新页面都要求你登录系统，这样的话相信没有几个用户会忍受的了。因此保存会话状态变得非常重要，于是cookie技术就产生了。

Cookie会根据从服务端发送的响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。当下次客户端再往该服务器发送请求时，客户端会自动在请求的报文中加入该Cookie值。

服务端接收到客户端发送的Cookie后，会去查找究竟是从哪一个客户端发来的连接请求。

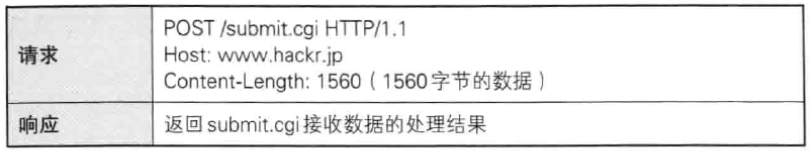


## 2.3 HTTP协议常用的方法

GET：用来获取资源，用来请求已被URL识别的资源。



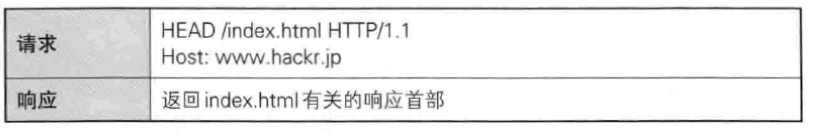
POST：用来传输实体主体



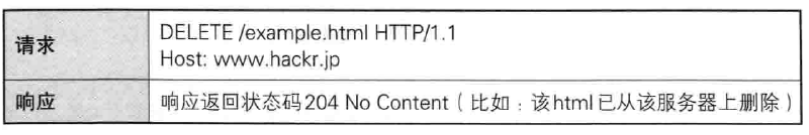
PUT：用来传输文件



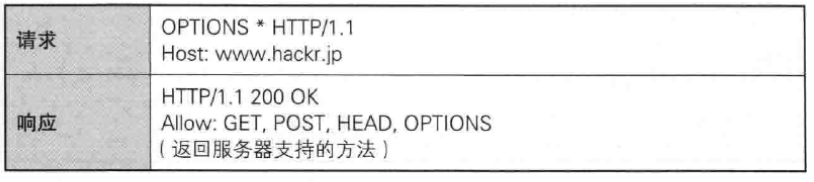
HEAD：用来获得报文首部



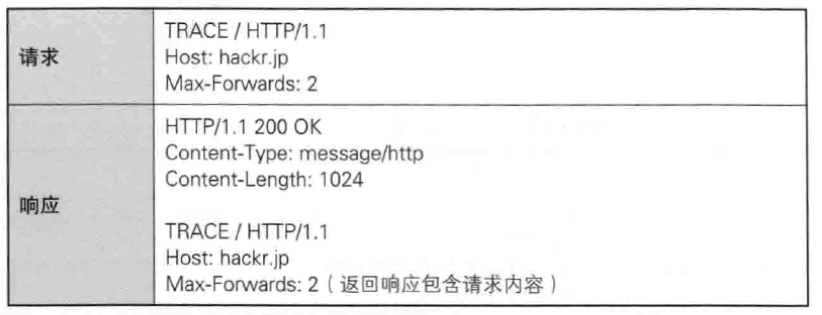
DELETE：用来删除文件



OPTIONS：用来询问支持的方法



TRACE：用来追踪路径



一般常用的为GET、POST、HEAD等，其他的了解即可。

POST和GET方法区别：

* **使用目标不同：**

POST与GET都用于获取信息，但是GET方式仅仅是查询，并不对服务器上的内容产生任何作用结果；每次GET的内容都是相同的。

POST则常用于发送一定的内容进行某些修改操作。

* **大小不同：**

由于不同的浏览器对URL的长度大小有一定的字符限制，因此由于GET方式放在URL的首部中，自然也跟着首先，但是具体的大小要依浏览器而定。

POST方式则是把内容放在报文内容中，因此只要报文的内容没有限制，它的大小就没有限制。

* **安全性不同：**

上面也说了GET是直接添加到URL后面的，直接就可以在URL中看到内容。而POST是放在报文内部的，用户无法直接看到。

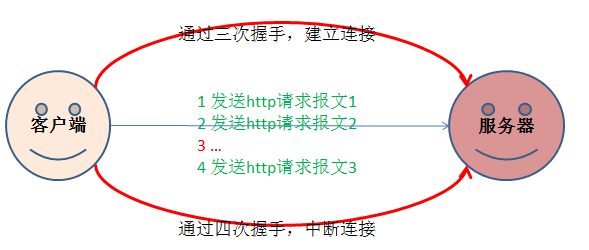
总的来说，GET用于获取某个内容，POST用于提交某种数据请求。

按照使用场景来说，一般用户注册的内容属于私密的，这应该使用POST方式；而针对某一内容的查询，为了快速的响应，可以使用GET方式。

## 2.4 持久连接节省通信量

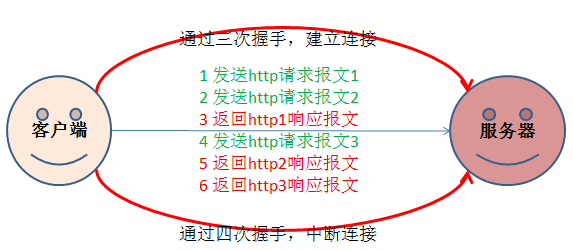
在HTTP协议的初始版本中，每进行一次HTTP通信就要断开TCP连接一次，在早起因为传输的都是文本数据，数据量较小没有太大影响，但是对于现在的网站包含有大量的图片，因此如果仍旧按照以前的方式去访问，这样就会造成通信量的消耗。为了解决该问题，HTTP 1.1提出了持久化理念，即只要任一端没有明确提出断开连接，则保持TCP连接状态。

持久连接的好处在于减少了TCP连接的重复建立和断开造成的额外开销，减轻了服务端的负载。在HTTP1.1中，所有请求默认都是持久连接。



## 2.5 管线化技术

持久连接使得多数请求以管线化方式发送成为可能，早期的版本中发送请求后需要等待并受到响应，才可以发送下一个请求。管线化技术的出现使得发送请求后不用等待响应，可直接发送下一次请求。



# 3、HTTP报文介绍

## 3.1 请求及响应报文的结构

请求行：包含请求的方法、请求的URL和HTTP协议版本；

状态行：包含表明结果的状态码、原因短语和HTTP协议版本；

首部字段：包含表示请求和响应的各种条件和属性的各类首部；

其他：包含HTTP的RFC里定义的首部（cookie）。

## 3.2 编码提升传输效率

HTTP在传输数据时可按照数据原貌进行传输，但是也可以在传输的过程中通过编码提升传输效率。一般包括以下几种编码方式：

* gzip(GUN zip)；
* conpress(UNIX系统的标准压缩)；
* deflate(zlib)；
* identity(不进行压缩)

## 3.3 发送多种数据的多部分对象集合

有的时候传输的内容，不仅仅是一些字符串，还有可能是一些图片，字符，音乐二进制等混杂的内容。

这就需要使用多部分对象集合，multipart，例如在使用java编写web上传文件的代码时，需要在form中指定form的编码格式。

设置form的enctype属性的值为multipart/form-data。

这是因为默认的情况下form使用的编码格式是：applicatin/x-www-form-urlencoded，这种编码格式会把所有的内容进行编码，不适合上传文件这种情况。

这两种编码格式的区别主要是：

* multipart/form-data 会以控件为基准，编码form中的内容。
* application/x-www-form-urlencoded 会把form中的内容编码成键值对的形式。

## 3.4 HTTP的范围请求

有些场景下，http报文请求一些很大的图片，但是加载过程很慢。

比如我们登录一些大图片的网址，会发现有时候图片是一块一块加载的。

这就是因为设置了http请求的长度，这样就可以分块的加载资源文件。

在请求报文中使用Range属性，在响应报文中使用Content-Type属性都可以指定一定字节范围的http请求。

# 4、返回结果的状态码

状态码的职责是当客户端向服务端发送请求时，描述返回的请求结果。借助状态码可以知道服务端是正常处理了请求，还是出现了错误。

* 1XX  提示信息 - 表示请求已被成功接收，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处理
* 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

# 5、其他内容

## 5.1 Cache 头域介绍

**If-Modified-Since**

作用：把浏览器端缓存页面的最后修改时间发送到服务器去，服务器会把这个时间与服务器上实际文件的最后修改时间进行对比。如果时间一致，那么返回304，客户端就直接使用本地缓存文件。如果时间不一致，就会返回200和新的文件内容。客户端接到之后，会丢弃旧文件，把新文件缓存起来，并显示在浏览器中。

**If-None-Match**

作用: If-None-Match和ETag一起工作，工作原理是在HTTP Response中添加ETag信息。 当用户再次请求该资源时，将在HTTP Request 中加入If-None-Match信息(ETag的值)。如果服务器验证资源的ETag没有改变（该资源没有更新），将返回一个304状态告诉客户端使用本地缓存文件。否则将返回200状态和新的资源和Etag。使用这样的机制将提高网站的性能。

看上述描述可能会觉得两个不是都一样的作用么，可为何又会同时存在呢？接下来看下两者的区别：

If-Modified-Since和 Last-Modified 一样都是用于记录页面最后修改时间的 HTTP 头信息，只是 Last-Modified 是由服务器往客户端发送的 HTTP 头，而 If-Modified-Since 则是由客户端往服务器发送的头，可 以看到，再次请求本地存在的 cache 页面时，客户端会通过 If-Modified-Since 头将先前服务器端发过来的 Last-Modified 最后修改时间戳发送回去，这是为了让服务器端进行验证，通过这个时间戳判断客户端的页面是否是最新的，如果不是最新的，则返回新的内容，如果是最新的，则 返回 304 告诉客户端其本地 cache 的页面是最新的，于是客户端就可以直接从本地加载页面了，这样在网络上传输的数据就会大大减少，同时也减轻了服务器的负担。

ETags和If-None-Match是一种常用的判断资源是否改变的方法。类似于Last-Modified和HTTP-IF-MODIFIED-SINCE。但是有所不同的是Last-Modified和HTTP-IF-MODIFIED-SINCE只判断资源的最后修改时间，而ETags和If-None-Match可以是资源任何的任何属性，不如资源的MD5等。

ETags和If-None-Match的工作原理是在HTTP Response中添加ETags信息。当客户端再次请求该资源时，将在HTTP Request中加入If-None-Match信息（ETags的值）。如果服务器验证资源的ETags没有改变（该资源没有改变），将返回一个304状态；否则，服务器将返回200状态，并返回该资源和新的ETags。

## 5.2 cookie和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。

如果说Cookie机制是通过检查客户身上的“通行证”来确定客户身份的话，那么Session机制就是通过检查服务器上的“客户明细表”来确认客户身份。Session相当于程序在服务器上建立的一份客户档案，客户来访的时候只需要查询客户档案表就可以了。

两者的区别：

* cookie数据存放在客户的浏览器上，session数据放在服务器上；
* cookie不是很安全，别人可以分析存放在本地的COOKIE并进行COOKIE欺骗，考虑到安全应当使用session；
* session会在一定时间内保存在服务器上。当访问增多，会比较占用你服务器的性能。考虑到减轻服务器性能方面，应当使用COOKIE；
* 单个cookie在客户端的限制是3K，就是说一个站点在客户端存放的COOKIE不能超过3K。