**TCP   HTTP   UDP:**

都是通信协议，也就是通信时所遵守的规则，只有双方按照这个规则“说话”，对方才能理解或为之服务。

**TCP   HTTP   UDP三者的关系:**

TCP/IP是个协议组，可分为四个层次：网络接口层、网络层、传输层和应用层。

在网络层有IP协议、ICMP协议、ARP协议、RARP协议和BOOTP协议。

在传输层中有TCP协议与UDP协议。

在应用层有FTP、HTTP、TELNET、SMTP、DNS等协议。

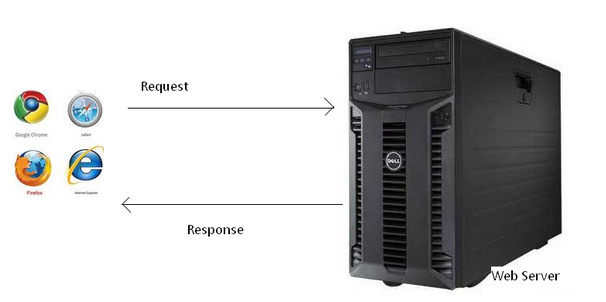
因此，HTTP本身就是一个协议，是从Web服务器传输超文本到本地浏览器的传送协议。

**socket:**  
这是为了实现以上的通信过程而建立成来的通信管道，其真实的代表是客户端和服务器端的一个通信进程，双方进程通过socket进行通信，而通信的规则采用指定的协议。socket只是一种连接模式，不是协议，tcp、udp，简单的说（虽然不准确）是两个最基本的协议,很多其它协议都是基于这两个协议如，http就是基于tcp的，.用socket可以创建tcp连接，也可以创建udp连接，这意味着，用socket可以创建任何协议的连接，因为其它协议都是基于此的。

下面我们主要来看一下和我们互联网生活密切相关的协议：HTTP

**什么是Http协议**

HTTP全称是HyperText Transfer Protocal，即：超文本传输协议，从1990年开始就在WWW上广泛应用，是现今在WWW上应用最多的协议，    Http是应用层协议，当你上网浏览网页的时候，浏览器和Web服务器之间就会通过HTTP在Internet上进行数据的发送和接收。Http是一个基于请求/响应模式的、无状态的协议。即我们通常所说的Request/Response。



**URL：**

URL(Uniform Resource Locator) 地址用于描述一个网络上的资源,  基本格式如下

schema://host[:port#]/path/…/[?query-string][#anchor]

scheme               指定低层使用的协议(例如：http, https, ftp)

host                   HTTP服务器的IP地址或者域名

port#                 HTTP服务器的默认端口是80，这种情况下端口号可以省略。如果使用了别的端口，必须指明，例如 http://www.cnblogs.com:8080/

path                   访问资源的路径

query-string       发送给http服务器的数据

anchor-             锚

URL 的一个例子

http://www.mywebsite.com/sj/test/test.aspx?name=sviergn&x=true#stuff

Schema:                 http  
host:                   www.mywebsite.com  
path:                   /sj/test/test.aspx  
Query String:           name=sviergn&x=true  
Anchor:                 stuff

**HTTP的Request/Response：**

先看Request 消息的结构,   Request 消息分为3部分

第一部分叫Request line,

 第二部分叫Request header,

第三部分是body. header和body之间有个空行，

 结构如下图

第一行中的Method表示请求方法,比如”POST”,”GET”,  Path-to-resoure表示请求的资源， Http/version-number 表示HTTP协议的版本号

当使用的是”GET” 方法的时候， body是为空的

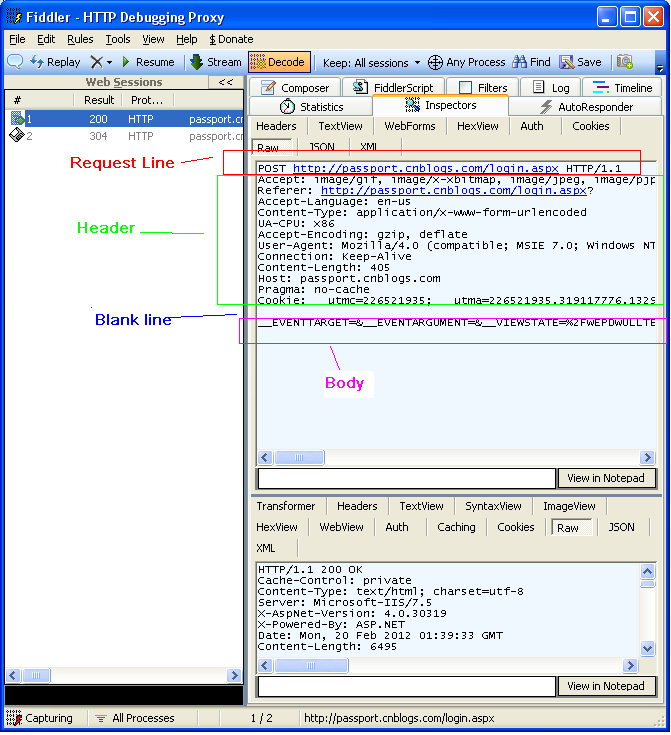
比如我们打开博客园首页的request 如下

GET http://www.cnblogs.com/ HTTP/1.1  
Host: [www.cnblogs.com](http://www.cnblogs.com/)

抽象的东西，难以理解，老感觉是虚的， 所谓眼见为实, 实际见到的东西，我们才能理解和记忆。 我们今天用Fiddler，实际的看看Request和Response.

下面我们打开Fiddler 捕捉一个博客园登录的Request 然后分析下它的结构, 在Inspectors tab下以Raw的方式可以看到完整的Request的消息，

 如下图



**Accept**

作用： 浏览器端可以接受的媒体类型,

例如：  Accept: text/html  代表浏览器可以接受服务器回发的类型为 text/html  也就是我们常说的html文档,

如果服务器无法返回text/html类型的数据,服务器应该返回一个406错误(non acceptable)

通配符 \* 代表任意类型

例如  Accept: \*/\*  代表浏览器可以处理所有类型,(一般浏览器发给服务器都是发这个)

**Referer:**

作用： 提供了Request的上下文信息的服务器，告诉服务器我是从哪个链接过来的，比如从我主页上链接到一个朋友那里，他的服务器就能够从HTTP Referer中统计出每天有多少用户点击我主页上的链接访问他的网站。

例如: Referer:http://translate.google.cn/?hl=zh-cn&tab=wT

**Accept-Language**

作用： 浏览器申明自己接收的语言。

语言跟字符集的区别：中文是语言，中文有多种字符集，比如big5，gb2312，gbk等等；

例如： Accept-Language: en-us

**Content-Type**

作用：

例如：Content-Type: application/x-www-form-urlencoded

**Accept-Encoding：**

作用： 浏览器申明自己接收的编码方法，通常指定压缩方法，是否支持压缩，支持什么压缩方法（gzip，deflate），（注意：这不是只字符编码）;

例如： Accept-Encoding: gzip, deflate

**User-Agent**

作用：告诉HTTP服务器， 客户端使用的操作系统和浏览器的名称和版本.

我们上网登陆论坛的时候，往往会看到一些欢迎信息，其中列出了你的操作系统的名称和版本，你所使用的浏览器的名称和版本，这往往让很多人感到很神奇，实际上，服务器应用程序就是从User-Agent这个请求报头域中获取到这些信息User-Agent请求报头域允许客户端将它的操作系统、浏览器和其它属性告诉服务器。

例如： User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; CIBA; .NET CLR 2.0.50727; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729; .NET4.0C; InfoPath.2; .NET4.0E)

**Connection**

例如：　Connection: keep-alive   当一个网页打开完成后，客户端和服务器之间用于传输HTTP数据的TCP连接不会关闭，如果客户端再次访问这个服务器上的网页，会继续使用这一条已经建立的连接

例如：  Connection: close  代表一个Request完成后，客户端和服务器之间用于传输HTTP数据的TCP连接会关闭， 当客户端再次发送Request，需要重新建立TCP连接。

**Content-Length**

作用：发送给HTTP服务器数据的长度。

例如： Content-Length: 38

**Host**（发送请求时，该报头域是必需的）

作用: 请求报头域主要用于指定被请求资源的Internet主机和端口号，它通常从HTTP URL中提取出来的

例如: 我们在浏览器中输入：http://www.guet.edu.cn/index.html

浏览器发送的请求消息中，就会包含Host请求报头域，如下：

Host：http://www.guet.edu.cn

此处使用缺省端口号80，若指定了端口号，则变成：Host：指定端口号

**Pragma**

作用： 防止页面被缓存， 在HTTP/1.1版本中，它和Cache-Control:no-cache作用一模一样

Pargma只有一个用法， 例如： Pragma: no-cache

**Cookie:**

作用： 最重要的header, 将cookie的值发送给HTTP 服务器

**Accept-Charset**

作用：浏览器申明自己接收的字符集，这就是本文前面介绍的各种字符集和字符编码，如gb2312，utf-8（通常我们说Charset包括了相应的字符编码方案）；

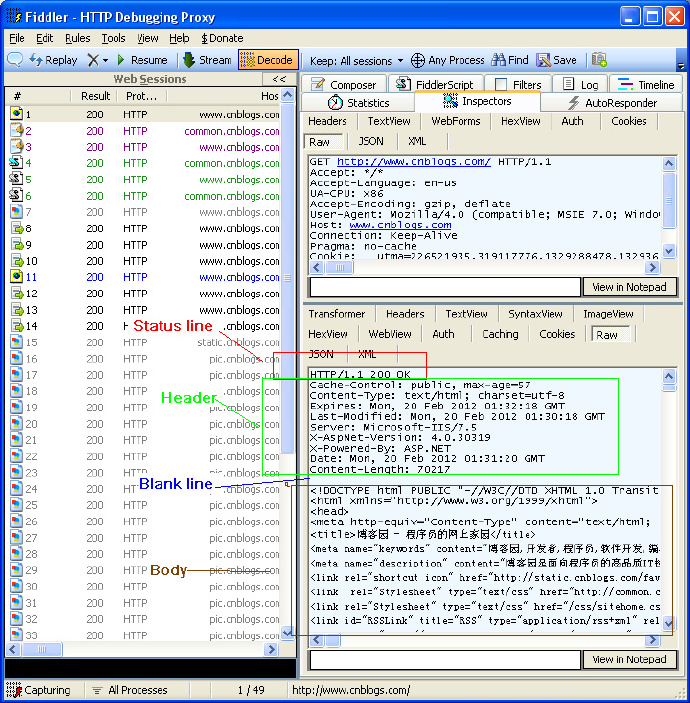
我们再看Response消息的结构, 和Request消息的结构基本一样。 同样也分为三部分

第一部分叫Response line,

 第二部分叫Response header，

第三部分是body. header和body之间也有个空行,

 结构如下图



HTTP/version-number表示HTTP协议的版本号，  status-code 和message 请看下节[[状态代码](http://www.cnblogs.com/TankXiao/archive/2012/02/13/2342672.html#statecode)]的详细解释.

我们用Fiddler 捕捉一个博客园首页的Response然后分析下它的结构, 在Inspectors tab下以Raw的方式可以看到完整的Response的消息，   如下图

**Cache-Control**

作用: 这个是非常重要的规则。 这个用来指定Response-Request遵循的缓存机制。各个指令含义如下

Cache-Control:Public   可以被任何缓存所缓存（）

Cache-Control:Private     内容只缓存到私有缓存中

Cache-Control:no-cache  所有内容都不会被缓存

还有其他的一些用法， 我没搞懂其中的意思， 请大家参考其他的资料

**Content-Type**

作用：WEB服务器告诉浏览器自己响应的对象的类型和字符集,

例如:

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

Content-Type:text/html;charset=GB2312

Content-Type: image/jpeg

**Expires**

作用: 浏览器会在指定过期时间内使用本地缓存

例如: Expires: Tue, 08 Feb 2022 11:35:14 GMT

**Last-Modified:**

作用： 用于指示资源的最后修改日期和时间。（实例请看上节的If-Modified-Since的实例）

例如: Last-Modified: Wed, 21 Dec 2011 09:09:10 GMT

**Server:**

作用：指明HTTP服务器的软件信息

例如:Server: Microsoft-IIS/7.5

**X-AspNet-Version:**

作用：如果网站是用ASP.NET开发的，这个header用来表示ASP.NET的版本

例如: X-AspNet-Version: 4.0.30319

**X-Powered-By:**

作用：表示网站是用什么技术开发的

例如： X-Powered-By: ASP.NET

**Connection**

例如：　Connection: keep-alive   当一个网页打开完成后，客户端和服务器之间用于传输HTTP数据的TCP连接不会关闭，如果客户端再次访问这个服务器上的网页，会继续使用这一条已经建立的连接

例如：  Connection: close  代表一个Request完成后，客户端和服务器之间用于传输HTTP数据的TCP连接会关闭， 当客户端再次发送Request，需要重新建立TCP连接。

**Content-Length**

指明实体正文的长度，以字节方式存储的十进制数字来表示。在数据下行的过程中，Content-Length的方式要预先在服务器中缓存所有数据，然后所有数据再一股脑儿地发给客户端。

例如: Content-Length: 19847

**Date**

作用:  生成消息的具体时间和日期

例如：　Date: Sat, 11 Feb 2012 11:35:14 GMT

**HTTP协议之Get和Post**

Http协议定义了很多与服务器交互的方法，最基本的有4种，分别是GET,POST,PUT,DELETE. 一个URL地址用于描述一个网络上的资源，而HTTP中的GET, POST, PUT, DELETE就对应着对这个资源的查，改，增，删4个操作。 我们最常见的就是GET和POST了。GET一般用于获取/查询资源信息，而POST一般用于更新资源信息.

我们看看GET和POST的区别

1. 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. GET提交的数据大小有限制（因为浏览器对URL的长度有限制），而POST方法提交的数据没有限制.

3. GET方式需要使用Request.QueryString来取得变量的值，而POST方式通过Request.Form来获取变量的值，也就是说Get是通过地址栏来传值，而Post是通过提交表单来传值。

4. GET方式提交数据，会带来安全问题，比如一个登录页面，通过GET方式提交数据时，用户名和密码将出现在URL上，如果页面可以被缓存或者其他人可以访问这台机器，就可以从历史记录获得该用户的账号和密码.