<http://www.cnblogs.com/EricaMIN1987_IT/p/3837436.html>

**这篇文章很好的解释了Http**

一、

http1.0协议默认的是非持久连接，

HTTP1.1默认的连接方式为持久连接。

非持久连接：每次服务器发出一个对象后，相应的TCP连接就被关闭，也就是说每个连接都没有持续到可用于传送其他对象。每个TCP连接只用于传输一个请求消息和一个响应消息。

持久连接：服务器在发出响应后让TCP连接继续打开着。同一对客户/服务器之间的后续请求和响应可以通过这个连接发送。HTTP/1.1的默认模式使用带流水线的持久连接。

二、http通信的实现有多种API，比如可以用java http模块的urlconnection 或者用apache 的http模块Httpclient（android的http模块用的是httpclient）。以下是两种不同API在功能上的一个比较。

|  |  |  |
| --- | --- | --- |
|  | **URLConnection** | **HTTPClient** |
| **Proxies and SOCKS** | Full support in Netscape browser, appletviewer, and applications (SOCKS: Version 4 only); no additional limitations from security policies. | Full support (SOCKS: Version 4 and 5); limited in applets however by security policies; in Netscape can't pick up the settings from the browser. |
| **Authorization** | Full support for Basic Authorization in Netscape (can use info given by the user for normal accesses outside of the applet); no support in appletviewer or applications. | Full support everywhere; however cannot access previously given info from Netscape, thereby possibly requesting the user to enter info (s)he has already given for a previous access. Also, you can add/implement additional authentication mechanisms yourself. |
| **Methods** | Only has GET and POST. | Has HEAD, GET, POST, PUT, DELETE, TRACE and OPTIONS, plus any arbitrary method. |
| **Headers** | Currently you can only set any request headers if you are doing a POST under Netscape; for GETs and the JDK you can't set any headers.  Under Netscape 3.0 you can read headers only if the resource was returned with a Content-length header; if no Content-length header was returned, or under previous versions of Netscape, or using the JDK no headers can be read. | Allows any arbitrary headers to be sent and received. |
| **Automatic Redirection Handling** | Yes. | Yes (as allowed by the HTTP/1.1 spec). |
| **Persistent Connections** | No support currently in JDK; under Netscape uses HTTP/1.0 Keep-Alive's. | Supports HTTP/1.0 Keep-Alive's and HTTP/1.1 persistence. |
| **Pipelining of Requests** | No. | Yes. |
| **Can handle protocols other than HTTP** | Theoretically; however only http is currently implemented. | No. |
| **Can do HTTP over SSL (https)** | Under Netscape, yes. Using Appletviewer or in an application, no. | No (not yet). |
| **Source code available** | No. | Yes. |

**三、**

**Get是向服务器发索取数据的一种请求，而Post是向服务器提交数据的一种请求，在FORM（表单）中，Method默认为"GET"，实质上，GET和POST只是发送机制不同，并不是一个取一个发！**

get：用于请求，以及信息获取，以明文方式传输，不改变服务器上的信息  
post：可能改变服务器上的资源请求，不以明文方式传输，安全性高

四、

<http://www.cnblogs.com/EricaMIN1987_IT/p/3837436.html>

这篇文章详细的介绍了http请求的格式 和格式中可以定义的属性及其值，响应格式和格式中可以定义的属性和值。

对于post命令：

报文分四个部分：请求行，请求头标，空行和请求数据

1）请求行  
  
请求行由三个标记组成：请求方法、请求URL和HTTP版本，中间用空格分开

例如： GET cuishen.iteye.com/blog/242842 HTTP/1.1

HTTP规范定义了8种可能的请求方法：(最常见的就是 GET 和 POST 两种方法)

* GET -- 检索URI中标识资源的一个简单请求
* HEAD -- 与GET方法相同，服务器只返回状态行和头标，并不返回请求文档
* POST -- 服务器接受被写入客户端输出流中的数据的请求
* PUT -- 服务器保存请求数据作为指定URI新内容的请求
* DELETE -- 服务器删除URI中命名的资源的请求
* OPTIONS -- 关于服务器支持的请求方法信息的请求
* TRACE -- Web服务器反馈Http请求和其头标的请求
* CONNECT -- 已文档化但当前未实现的一个方法，预留做隧道处理

2）请求头标

请求头标：由key ：value 健值组成，每行一对。请求头标用来通知服务器有关客户端的功能和标识。

HOST -- 请求的哪一个服务器端地址，主地址，比如：我的技术blog：cuishen.iteye.com  
User-Agent -- 用户即客户端可以使用的浏览器 ，如： Mozilla/4.0  
Accept -- 即客户端可以接受的MIME 类型列表，如image/gif、text/html、application/msword  
Content-Length -- 只适用于POST请求，以字节给出POST数据的尺寸

3）空行

发送回车符和退行，通知服务器以下不再有头标。

4）请求数据

使用POST传送数据，最常使用的是Content-Type和Content-Length头标。

请求报文总结：  
  
我们可以这样写出一个标准的 HTTP请求：

|  |
| --- |
| POST /blog/242842 HTTP1.1 HOST: cuishen.iteye.com/ User-Agent: Mozilla/4.0 Accpt: image/gif,text/html,application/pdf,image/png... key=value&key=value&key=value...... (POST()请求的数据) |

这上面的一个例子意思是：

|  |
| --- |
| 我要去访问的服务器端的地址是cuishen.iteye.com/ 它下面的资源 /blog/242842 连起来就是： cuishen.iteye.com/blog/242842 这个页面用的是 HTTP1.1 规范，我的浏览器版本是Mozilla/4.0  可以支持的MIME格式为 image/gif,text/html,application/pdf,image/png...等等 |

|  |
| --- |
| 这个MIME格式我们在servlet中写法是：response.setContentType("text/html;charset=gb2312"); 或者在jsp中写法是：<%@ page contentType="text/html;charset=gb2312"%> 或者在html中写法是：<meta http-equiv="content-Type" content="text/html; charset=gb2312"> |

GET 和 POST 最直观的区别就是：GET方法将数据的请求跟在了所请求的URL后面，也就是在请求行里面我们是这么样来做的：

1.           GET /blog/242842?key=value&key=value&key=value......HTTP1.1

实际上用 GET 是这样传递数据的：

1.           http://cuishen.iteye.com/?page=2......

iii.服务器端响应请求生成结果并回发（response）

Web 服务器解析请求，定位指定的资源 http://cuishen.iteye.com/blog/242842

1）根据请求时的 GET/POST 对应的用servlet里的 doGet() / doPost()方法来处理（有可能是一些业务逻辑，也有可能是一些验证等等，也有可能是一些数据查询，提交等等）其有效的数据就来源于key=value&key=value&key=value......，以及其它的一些封装在 request 对象中的数据资源。

2）处理请求之后，由 response 对象得到 java.io.PrintWriter 输出流对象out，通过 out.println(); 将数据以指定的格式，如按照response.setcontentType("text/html;charset=gb2312");的格式输出到输出流。

它的响应报文与请求报文非常类似，其区别就在于：我们在请求阶段的请求行被状态行给替换了，再来看响应报文：

3）一个响应报文由四个部分组成：状态行、响应头标、空行、响应数据：

(a).状态行：

状态行由三个标记组成：HTTP版本、响应代码和响应描述。

|  |
| --- |
| HTTP1.1 --- 100 --- continue //继续追加后继内容 HTTP1.1 --- 200 --- OK //一切正常 HTTP1.1 --- 301 --- Moved Permanently //请求的文档在其它地方，会自动连接 HTTP1.1 --- 403 --- Forbidden //绝对拒绝你访问这个资源，不管授权没有 HTTP1.1 --- 400 --- Bad Request //客户端请求中的不良语法 HTTP1.1 --- 404 --- Not Found //最常见，绝对是大名鼎鼎的找不到 |

HTTP响应码：

|  |
| --- |
| 1xx：提示性信息，告诉客户端应该对某些其它的动作作出响应 2xx：这些就代表了请求成功 3xx：重定向，为了完成请求，必须进一步执行的动作 4xx：客户端错误 500-599: 服务器端的错误 |

(b).响应头标：像请求头标一样，它们指出服务器的功能，标识出响应数据的细节。

|  |
| --- |
| Date: Sat, 31 Dec 2005 23:59:59 GMT --响应生成的日期和时间 ContentType: 'text/html;charset=gb2312' Content-Length: 122 --响应中的字节数，只在浏览器使用永久（Keep-alive）HTTP连接时需要。 |

(c).空行：最后一个响应头标之后是一个空行，发送回车符和退行，表明服务器以下不再有头标。

(d).响应数据：HTML文档和图像等，也就是HTML本身。out.println("<html>......");写到客户端。

1.           **<html>**

2.           **<head>**

3.           **<title>**Welcome to cuishen's IT blog**</title>**

4.           **</head>**

5.           **<body>**

6.           <!-- 这里是具体的内容，看到了这里

7.           相信大家对 HTTP 工作原理及客户端与服务器交互过程已经很清楚了吧

8.           --**>**

9.           **</body>**

10.       **</html>**