引言
HTTP 是一个属于应用层的面向对象的协议，由于其简捷、快速的方式，适用于分布式超媒体信息系统。它于 1990 年提出，经过几年的使用与发展，得到不断地完善和 扩展。目前在 WWW 中使用的是 HTTP/1.0 的第六版，HTTP/1.1 的规范化工作正在进行之中，而且 HTTP-NG(Next Generation of HTTP)的建议已经提出。

HTTP 协议的主要特点可概括如下：
1.支持客户/服务器模式。
2.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。请求方法常用的有 GET、HEAD、POST。每种方法规定了客户与服务器联系的类型不同。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。
3.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。
4.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。
5.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。

一、HTTP 协议详解之 URL 篇
　　http（超文本传输协议）是一个基于请求与响应模式的、无状态的、应用层的协议，常基于 TCP 的连接方式，HTTP1.1 版本中给出一种持续连接的机制，绝大多数的 Web 开发，都是构建在 HTTP 协议之上的 Web 应用。
HTTP URL (URL 是一种特殊类型的 URI，包含了用于查找某个资源的足够的信息)的格式如下：
http://host[":"port][abs_path]
http 表示要通过 HTTP 协议来定位网络资源；host 表示合法的 Internet 主机域名或者 IP 地址；port 指定一个端口号，为空则使用缺省端口 80；abs_path 指定请求资源的 URI；如果 URL 中没有给出 abs_path，那么当它作为请求 URI 时，必须以“/”的形式给出，通常这个工作 浏览器自动帮我们完成。
eg:
1、输入：www.guet.edu.cn
浏览器自动转换成：http://www.guet.edu.cn/
2、http:192.168.0.116:8080/index.jsp

二、HTTP 协议详解之请求篇
　　http 请求由三部分组成，分别是：请求行、消息报头、请求正文
1、请求行以一个方法符号开头，以空格分开，后面跟着请求的 URI 和协议的版本，格式如下：Method Request-URI HTTP-Version CRLF
其中 Method 表示请求方法；Request-URI 是一个统一资源标识符；HTTP-Version 表示请求的 HTTP 协议版本；CRLF 表示回车和换行（除了作为结尾的 CRLF 外，不允许出现单独的 CR 或 LF 字符）。
请求方法（所有方法全为大写）有多种，各个方法的解释如下：
GET　　请求获取 Request-URI 所标识的资源
POST　 在 Request-URI 所标识的资源后附加新的数据
HEAD　 请求获取由 Request-URI 所标识的资源的响应消息报头
PUT　　请求服务器存储一个资源，并用 Request-URI 作为其标识
DELETE　请求服务器删除 Request-URI 所标识的资源
TRACE　请求服务器回送收到的请求信息，主要用于测试或诊断
CONNECT 保留将来使用
OPTIONS 请求查询服务器的性能，或者查询与资源相关的选项和需求
应用举例：
GET 方法：在浏览器的地址栏中输入网址的方式访问网页时，浏览器采用 GET 方法向服务器获取资源，eg:GET /form.html HTTP/1.1 (CRLF)
POST 方法要求被请求服务器接受附在请求后面的数据，常用于提交表单。
eg：POST /reg.jsp HTTP/ (CRLF)
Accept:image/gif,image/x-xbit,... (CRLF)
...
HOST:www.guet.edu.cn (CRLF)
Content-Length:22 (CRLF)
Connection:Keep-Alive (CRLF)
Cache-Control:no-cache (CRLF)
(CRLF)　　//该 CRLF 表示消息报头已经结束，在此之前为消息报头
user=jeffrey&pwd=1234　//此行以下为提交的数据
HEAD 方法与 GET 方法几乎是一样的，对于 HEAD 请求的回应部分来 说，它的 HTTP 头部中包含的信息与通过 GET 请求所得到的信息是相同的。利用这个方法，不必传输整个资源内容，就可以得到 Request-URI 所标识 的资源的信息。该方法常用于测试超链接的有效性，是否可以访问，以及最近是否更新。

2、请求报头后述
3、请求正文(略)

三、HTTP 协议详解之响应篇
　在接收和解释请求消息后，服务器返回一个 HTTP 响应消息。
HTTP 响应也是由三个部分组成，分别是：状态行、消息报头、响应正文
1、状态行格式如下：
HTTP-Version Status-Code Reason-Phrase CRLF
其中，HTTP-Version 表示服务器 HTTP 协议的版本；Status-Code 表示服务器发回的响应状态代码；Reason-Phrase 表示状态代码的文本描述。
状态代码有三位数字组成，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
常见状态代码、状态描述、说明：
200 OK　　//客户端请求成功
400 Bad Request　//客户端请求有语法错误，不能被服务器所理解
401 Unauthorized //请求未经授权，这个状态代码必须和 WWW-Authenticate 报头域一起使用
403 Forbidden　//服务器收到请求，但是拒绝提供服务
404 Not Found　//请求资源不存在，eg：输入了错误的 URL
500 Internal Server Error //服务器发生不可预期的错误
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
eg：HTTP/1.1 200 OK （CRLF）
2、响应报头后述
3、响应正文就是服务器返回的资源的内容

四、HTTP 协议详解之消息报头篇
　HTTP 消息由客户端到服务器的请求和服务器到客户端的响应组成。请求消息和响应消息都是由开始行（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。
HTTP 消息报头包括普通报头、请求报头、响应报头、实体报头。
每一个报头域都是由名字+“：”+空格+值 组成，消息报头域的名字是大小写无关的。
1、普通报头
在普通报头中，有少数报头域用于所有的请求和响应消息，但并不用于被传输的实体，只用于传输的消息。
eg：
Cache-Control　用于指定缓存指令，缓存指令是单向的（响应中出现的缓存指令在请求中未必会出现），且是独立的（一个消息的缓存指令不会影响另一个消息处理的缓存机制），HTTP1.0 使用的类似的报头域为 Pragma。
请求时的缓存指令包括：no-cache（用于指示请求或响应消息不能缓存）、no-store、max-age、max-stale、min-fresh、only-if-cached;
响应时的缓存指令包括：public、private、no-cache、no-store、no-transform、must-revalidate、proxy-revalidate、max-age、s-maxage.
eg：为了指示 IE 浏览器（客户端）不要缓存页面，服务器端的 JSP 程序可以编写如下：response.sehHeader("Cache-Control","no-cache");
//response.setHeader("Pragma","no-cache");作用相当于上述代码，通常两者//合用
这句代码将在发送的响应消息中设置普通报头域：Cache-Control:no-cache

Date 普通报头域表示消息产生的日期和时间
Connection 普通报头域允许发送指定连接的选项。例如指定连接是连续，或者指定“close”选项，通知服务器，在响应完成后，关闭连接
2、请求报头
请求报头允许客户端向服务器端传递请求的附加信息以及客户端自身的信息。
常用的请求报头
Accept
Accept 请求报头域用于指定客户端接受哪些类型的信息。eg：Accept：image/gif，表明客户端希望接受 GIF 图象格式的资源；Accept：text/html，表明客户端希望接受 html 文本。
Accept-Charset
Accept-Charset 请求报头域用于指定客户端接受的字符集。eg：Accept-Charset:iso-8859-1,gb2312.如果在请求消息中没有设置这个域，缺省是任何字符集都可以接受。
Accept-Encoding

Accept-Encoding 请求报头域类似于 Accept，但是它是用于指定可接受的内容编码。eg：Accept-Encoding:gzip.deflate.如果请求消息中没有设置这个域服务器假定客户端对各种内容编码都可以接受。
Accept-Language
Accept-Language 请求报头域类似于 Accept，但是它是用于指定一种自然语言。eg：Accept-Language:zh-cn.如果请求消息中没有设置这个报头域，服务器假定客户端对各种语言都可以接受。
Authorization
Authorization 请求报头域主要用于证明客户端有权查看某个资源。当浏览器访问一个页面时，如果收到服务器的响应代码为 401（未授权），可以发送一个包含 Authorization 请求报头域的请求，要求服务器对其进行验证。
Host（发送请求时，该报头域是必需的）
Host 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，eg：
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：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：www.guet.edu.cn:指定端口号
User-Agent
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。不过，这个报头域不是必需的，如果我们自己编写一个浏览器，不使用 User-Agent 请求报头域，那么服务器端就无法得知我们的信息 了。
请求报头举例：
GET /form.html HTTP/1.1 (CRLF)
Accept:image/gif,image/x-xbitmap,image/jpeg,application/x-shockwave-flash,application/vnd.ms-excel,application/vnd.ms-powerpoint,application/msword,*/* (CRLF)
Accept-Language:zh-cn (CRLF)
Accept-Encoding:gzip,deflate (CRLF)
If-Modified-Since:Wed,05 Jan 2007 11:21:25 GMT (CRLF)
If-None-Match:W/"80b1a4c018f3c41:8317" (CRLF)
User-Agent:Mozilla/4.0(compatible;MSIE6.0;Windows NT 5.0) (CRLF)
Host:www.guet.edu.cn (CRLF)
Connection:Keep-Alive (CRLF)
(CRLF)
3、响应报头
响应报头允许服务器传递不能放在状态行中的附加响应信息，以及关于服务器的信息和对 Request-URI 所标识的资源进行下一步访问的信息。
常用的响应报头
Location
Location 响应报头域用于重定向接受者到一个新的位置。Location 响应报头域常用在更换域名的时候。
Server
Server 响应报头域包含了服务器用来处理请求的软件信息。与 User-Agent 请求报头域是相对应的。下面是 Server 响应报头域的一个例子：
Server：Apache-Coyote/1.1
WWW-Authenticate
WWW-Authenticate 响应报头域必须被包含在 401（未授权的）响应消息中，客户端收到 401 响应消息时候，并发送 Authorization 报头域请求服务器对其进行验证时，服务端响应报头就包含该报头域。
eg：WWW-Authenticate:Basic realm="Basic Auth Test!" //可以看出服务器对请求资源采用的是基本验证机制。

4、实体报头
请求和响应消息都可以传送一个实体。一个实体由实体报头域和实体正文组成，但并不是说实体报头域和实体正文要在一起发送，可以只发送实体报头域。实体报头定义了关于实体正文（eg：有无实体正文）和请求所标识的资源的元信息。
常用的实体报头
Content-Encoding
Content-Encoding 实体报头域被用作媒体类型的修饰符，它的值指示了已经被应用到实体正文的附加内容的编码，因而要获得 Content- Type 报头域中所引用的媒体类型，必须采用相应的解码机制。Content-Encoding 这样用于记录文档的压缩方法，eg：Content- Encoding：gzip
Content-Language
Content-Language 实体报头域描述了资源所用的自然语言。没有设置该域则认为实体内容将提供给所有的语言阅读

者。eg：Content-Language:da
Content-Length
Content-Length 实体报头域用于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。
Content-Type
Content-Type 实体报头域用语指明发送给接收者的实体正文的媒体类型。eg：
Content-Type:text/html;charset=ISO-8859-1
Content-Type:text/html;charset=GB2312
Last-Modified
Last-Modified 实体报头域用于指示资源的最后修改日期和时间。
Expires
Expires 实体报头域给出响应过期的日期和时间。为了让代理服务器或浏览器在一段时间以后更新缓存中(再次访问曾访问过的页面时，直接从缓存中加载， 缩短响应时间和降低服务器负载)的页面，我们可以使用 Expires 实体报头域指定页面过期的时间。eg：Expires：Thu，15 Sep 2006 16:23:12 GMT
HTTP1.1 的客户端和缓存必须将其他非法的日期格式（包括 0）看作已经过期。eg：为了让浏览器不要缓存页面，我们也可以利用 Expires 实体报头域，设置为 0，jsp 中程序如下：response.setDateHeader("Expires","0");

五、利用 telnet 观察 http 协议的通讯过程
实验目的及原理：
利用 MS 的 telnet 工具，通过手动输入 http 请求信息的方式，向服务器发出请求，服务器接收、解释和接受请求后，会返回一个响应，该响应会在 telnet 窗口上显示出来，从而从感性上加深对 http 协议的通讯过程的认识。
实验步骤：
1、打开 telnet
1.1 打开 telnet
运行-->cmd-->telnet
1.2 打开 telnet 回显功能
set localecho
2、连接服务器并发送请求
2.1 open www.guet.edu.cn 80 //注意端口号不能省略
HEAD /index.asp HTTP/1.0
Host:www.guet.edu.cn

/*我们可以变换请求方法,请求桂林电子主页内容,输入消息如下*/
open www.guet.edu.cn 80

GET /index.asp HTTP/1.0 //请求资源的内容
Host:www.guet.edu.cn
2.2 open www.sina.com.cn 80 //在命令提示符号下直接输入 telnet www.sina.com.cn 80
HEAD /index.asp HTTP/1.0
Host:www.sina.com.cn

3 实验结果：
3.1 请求信息 2.1 得到的响应是:
HTTP/1.1 200 OK //请求成功
Server: Microsoft-IIS/5.0 //web 服务器
Date: Thu,08 Mar 200707:17:51 GMT
Connection: Keep-Alive
Content-Length: 23330
Content-Type: text/html
Expries: Thu,08 Mar 2007 07:16:51 GMT
Set-Cookie: ASPSESSIONIDQAQBQQQB=BEJCDGKADEDJKLKKAJEOIMMH; path=/
Cache-control: private
//资源内容省略
3.2 请求信息 2.2 得到的响应是:
HTTP/1.0 404 Not Found //请求失败
Date: Thu, 08 Mar 2007 07:50:50 GMT
Server: Apache/2.0.54 <Unix>
Last-Modified: Thu, 30 Nov 2006 11:35:41 GMT
ETag: "6277a-415-e7c76980"
Accept-Ranges: bytes
X-Powered-By: mod_xlayout_jh/0.0.1vhs.markII.remix

Vary: Accept-Encoding
Content-Type: text/html
X-Cache: MISS from zjm152-78.sina.com.cn
Via: 1.0 zjm152-78.sina.com.cn:80<squid/2.6.STABLES-20061207>
X-Cache: MISS from th-143.sina.com.cn
Connection: close

失去了跟主机的连接
按任意键继续...
4 .注意事项：1、出现输入错误，则请求不会成功。
2、报头域不分大小写。
3、更深一步了解 HTTP 协议，可以查看 RFC2616，在 http://www.letf.org/rfc 上找到该文件。
4、开发后台程序必须掌握 http 协议
六、HTTP 协议相关技术补充
1、基础：
高层协议有：文件传输协议 FTP、电子邮件传输协议 SMTP、域名系统服务 DNS、网络新闻传输协议 NNTP 和 HTTP 协议等
中介由三种：代理(Proxy)、网关(Gateway)和通道(Tunnel)，一个代理根据 URI 的绝对格式来接受请求，重写全部或部分消息，通过 URI 的标识把已格式化过的请求发送到服务器。网关是一个接收代理，作为一些其它服务器的上层，并且如果必须的话，可以把请求翻译给下层的服务器协议。一 个通道作为不改变消息的两个连接之间的中继点。当通讯需要通过一个中介(例如：防火墙等)或者是中介不能识别消息的内容时，通道经常被使用。
代理(Proxy)：一个中间程序，它可以充当一个服务器，也可以充当一个客户机，为其它客户机建立请求。请求是通过可能的翻译在内部或经过传递到其它的 服务器中。一个代理在发送请求信息之前，必须解释并且如果可能重写它。代理经常作为通过防火墙的客户机端的门户，代理还可以作为一个帮助应用来通过协议处理没有被用户代理完成的请求。
网关(Gateway)：一个作为其它服务器中间媒介的服务器。与代理不同的是，网关接受请求就好象对被请求的资源来说它就是源服务器；发出请求的客户机并没有意识到它在同网关打交道。
网关经常作为通过防火墙的服务器端的门户，网关还可以作为一个协议翻译器以便存取那些存储在非 HTTP 系统中的资源。
通道(Tunnel)：是作为两个连接中继的中介程序。一旦激活，通道便被认为不属于 HTTP 通讯，尽管通道可能是被一个 HTTP 请求初始化的。当被中继 的连接两端关闭时，通道便消失。当一个门户(Portal)必须存在或中介(Intermediary)不能解释中继的通讯时通道被经常使用。

2、协议分析的优势—HTTP 分析器检测网络攻击
以模块化的方式对高层协议进行分析处理，将是未来入侵检测的方向。
HTTP 及其代理的常用端口 80、3128 和 8080 在 network 部分用 port 标签进行了规定

3、HTTP 协议 Content Lenth 限制漏洞导致拒绝服务攻击
使用 POST 方法时，可以设置 ContentLenth 来定义需要传送的数据长度，例如 ContentLenth:999999999，在传送完成前，内 存不会释放，攻击者可以利用这个缺陷，连续向 WEB 服务器发送垃圾数据直至 WEB 服务器内存耗尽。这种攻击方法基本不会留下痕迹。
http://www.cnpaf.net/Class/HTTP/0532918532667330.html

4、利用 HTTP 协议的特性进行拒绝服务攻击的一些构思
服务器端忙于处理攻击者伪造的 TCP 连接请求而无暇理睬客户的正常请求（毕竟客户端的正常请求比率非常之小），此时从正常客户的角度看来，服务器失去响应，这种情况我们称作：服务器端受到了 SYNFlood 攻击（SYN 洪水攻击）。
而 Smurf、TearDrop 等是利用 ICMP 报文来 Flood 和 IP 碎片攻击的。本文用“正常连接”的方法来产生拒绝服务攻击。
19 端口在早期已经有人用来做 Chargen 攻击了，即 Chargen_Denial_of_Service，但是！他们用的方法是在两台 Chargen 服务器之间产生 UDP 连接，让服务器处理过多信息而 DOWN 掉，那么，干掉一台 WEB 服务器的条件就必须有 2 个：1.有 Chargen 服务 2.有 HTTP 服务
方法：攻击者伪造源 IP 给 N 台 Chargen 发送连接请求（Connect），Chargen 接收到连接后就会返回每秒 72 字节的字符流（实际上根据网络实际情况，这个速度更快）给服务器。

5、Http 指纹识别技术
Http 指纹识别的原理大致上也是相同的：记录不同服务器对 Http 协议执行中的微小差别进行识别.Http 指纹识别比 TCP/IP 堆栈指纹识别复杂许 多,理由是定制 Http 服务器的配置文件、增加插件或组件使得更改 Http 的响应信息变的很容易,这样使得识别变的困难；然而定制 TCP/IP 堆栈的行为 需要对核心层进行修改,所以就容易识别.
要让服务器返回不同的 Banner 信息的设置是很简单的,象 Apache 这样的开放源代码的 Http 服务器,用

户可以在源代码里修改 Banner 信息,然 后重起 Http 服务就生效了；对于没有公开源代码的 Http 服务器比如微软的 IIS 或者是 Netscape,可以在存放 Banner 信息的 Dll 文件中修 改,相关的文章有讨论的,这里不再赘述,当然这样的修改的效果还是不错的.另外一种模糊 Banner 信息的方法是使用插件。
常用测试请求：
1：HEAD/Http/1.0 发送基本的 Http 请求
2：DELETE/Http/1.0 发送那些不被允许的请求,比如 Delete 请求
3：GET/Http/3.0 发送一个非法版本的 Http 协议请求
4：GET/JUNK/1.0 发送一个不正确规格的 Http 协议请求
Http 指纹识别工具 Httprint,它通过运用统计学原理,组合模糊的逻辑学技术,能很有效的确定 Http 服务器的类型.它可以被用来收集和分析不同 Http 服务器产生的签名。

6、其他：为了提高用户使用浏览器时的性能，现代浏览器还支持并发的访问方式，浏览一个网页时同时建立多个连接，以迅速获得一个网页上的多个图标，这样能更快速完成整个网页的传输。
HTTP1.1 中提供了这种持续连接的方式，而下一代 HTTP 协议：HTTP-NG 更增加了有关会话控制、丰富的内容协商等方式的支持，来提供
更高效率的连接。

HTTP 协议的头信息详解
通常 HTTP 消息包括客户机向服务器的请求消息和服务器向客户机的响应消息。这两种类型的消息由一个起始行，一个或者多个头域，一个只是头域结束的 空行和可 选的消息体组成。HTTP 的头域包括通用头，请求头，响应头和实体头四个部分。每个头域由一个域名，冒号（:）和域值三部分组成。域名是大小写无关的，域 值前可以添加任何数量的空格符，头域可以被扩展为多行，在每行开始处，使用至少一个空格或制表符。

通用头域

通用头 域包含请求和响应消息都支持的头域，通用头域包含 Cache-Control、 Connection、Date、Pragma、Transfer-Encoding、Upgrade、Via。对通用头域的扩展要求通讯双方都支持此扩 展，如果存在不支持的通用头域，一般将会作为实体头域处理。下面简单介绍几个在 UPnP 消息中使用的通用头域。

Cache-Control 头域

Cache -Control 指定请求和响应遵循的缓存机制。在请求消息或响应消息中设置 Cache-Control 并不会修改另一个消息处理过程中的缓存处理过程。请求时的缓存指令包括 no-cache、no-store、max-age、max-stale、min-fresh、only-if-cached，响应消息中的指令包括 public、private、no-cache、no- store、no-transform、must-revalidate、proxy-revalidate、max-age。各个消息中的指令含义如 下：

Public 指示响应可被任何缓存区缓存。

Private 指示对于单个用户的整个或部分响应消息，不能被共享缓存处理。这允许服务器仅仅描述当用户的部分响应消息，此响应消息对于其他用户的请求无效。

no-cache 指示请求或响应消息不能缓存

no-store 用于防止重要的信息被无意的发布。在请求消息中发送将使得请求和响应消息都不使用缓存。

max-age 指示客户机可以接收生存期不大于指定时间（以秒为单位）的响应。

min-fresh 指示客户机可以接收响应时间小于当前时间加上指定时间的响应。

max-stale 指示客户机可以接收超出超时期间的响应消息。如果指定 max-stale 消息的值，那么客户机可以接收超出超时期指定值之内的响应消息。

Date 头域

Date 头域表示消息发送的时间，时间的描述格式由 rfc822 定义。例如，Date:Mon,31Dec200104:25:57GMT。Date 描述的时间表示世界标准时，换算成本地时间，需要知道用户所在的时区。

Pragma 头域

Pragma 头域用来包含实现特定的指令，最常用的是 Pragma:no-cache。在 HTTP/1.1 协议中，它的含义和 Cache- Control:no-cache 相同。

请求消息

请求消息的第一行为下面的格式：

MethodSPRequest-URISPHTTP-VersionCRLFMethod 表示对于 Request-URI 完成的方法，这个字段是大小写敏感的，包括 OPTIONS、GET、HEAD、POST、PUT、DELETE、 TRACE。方法 GET 和 HEAD 应该被所有的通用 WEB 服务器支持，其他所有方法的实现是可选的。GET 方法取回由 Request-URI 标识的信息。 HEAD 方法也是取回由 Request-URI 标识的信息，只是可以在响应时，不返回消息体。POST 方法可以请求服务器接收包含在请求中的实体信息，可 以用于提交表单，向新闻组、BBS、邮件群组和数据库发送消息。

SP 表示空格。Request-URI 遵循 URI 格式，在此字段 为星 号（*）时，说明请求并不用于某个特定的资源地址，而是用于服务器本身。HTTP- Version 表示支持的 HTTP 版本，例如为 HTTP/1.1。CRLF 表示换行回车符。请求头域允许客户端向服务器传递关于请求或者关于客户机的附加 信息。请求头域可能包含下列字段 Accept、Accept-Charset、Accept- Encoding、Accept-Language、Authorization、From、Host、If-Modified-Since、If- Match、If-None-Match、If-Range、If-Range、If-Unmodified-Since、Max-Forwards、 Proxy-Authorization、Range、Referer、User-Agent。对请求头域的扩展要求通讯双方都支持，如果存在不支持的请 求头域，一般将会作为实体头域处理。

典型的请求消息：

```
GET http://download.microtool.de:80/somedata.exe
Host: download.microtool.de
Accept:*/*
Pragma: no-cache
Cache-Control: no-cache
Referer: http://download.microtool.de/
User-Agent:Mozilla/4.04[en](Win95;I;Nav)
Range:bytes=554554-
```

上例第一行表示 HTTP 客户端（可能是浏览器、下载程序）通过 GET 方法获得指定 URL 下的文件。棕色的部分表示请求头域的信息，绿色的部分表示通用头部分。

Host 头域

Host 头域指定请求资源的 Intenet 主机和端口号，必须表示请求 url 的原始服务器或网关的位置。HTTP/1.1 请求必须包含主机头域，否则系统会以 400 状态码返回。

Referer 头域

Referer 头域允许客户端指定请求 uri 的源资源地址，这可以允许服务器生成回退链表，可用来登陆、优化 cache 等。他也允许废除的或错误的连接由于维护的目的被 追踪。如果请求的 uri 没有自己的 uri 地址，Referer 不能被发送。如果指定的是部分 uri 地址，则此地址应该是一个相对地址。

Range 头域

Range 头域可以请求实体的一个或者多个子范围。例如，
表示头 500 个字节：bytes=0-499
表示第二个 500 字节：bytes=500-999
表示最后 500 个字节：bytes=-500
表示 500 字节以后的范围：bytes=500-
第一个和最后一个字节：bytes=0-0,-1
同时指定几个范围：bytes=500-600,601-999

但是服务器可以忽略此请求头，如果无条件 GET 包含 Range 请求头，响应会以状态码 206（PartialContent）返回而不是以 200 （OK）。

User-Agent 头域

User-Agent 头域的内容包含发出请求的用户信息。

响应消息

响应消息的第一行为下面的格式：

HTTP-VersionSPStatus-CodeSPReason-PhraseCRLF

HTTP -Version 表示支持的 HTTP 版本，例如为 HTTP/1.1。Status- Code 是一个三个数字的结果代码。Reason-Phrase 给 Status-Code 提供一个简单的文本描述。Status-Code 主要用于机器自 动识别，Reason-Phrase 主要用于帮助用户理解。Status-Code 的第一个数字定义响应的类别，后两个数字没有分类的作用。第一个数字可 能取 5 个不同的值：

1xx:信息响应类，表示接收到请求并且继续处理

2xx:处理成功响应类，表示动作被成功接收、理解和接受

3xx:重定向响应类，为了完成指定的动作，必须接受进一步处理

4xx:客户端错误，客户请求包含语法错误或者是不能正确执行

5xx:服务端错误，服务器不能正确执行一个正确的请求

响应头域允许服务器传递不能放在状态行的附加信息，这些域主要描述服务器的信息和 Request-URI 进一步的信息。响应头域包含 Age、Location、Proxy-Authenticate、Public、Retry- After、Server、Vary、Warning、WWW-Authenticate。对响应头域的扩展要求通讯双方都支持，如果存在不支持的响应头 域，一般将会作为实体头域处理。

典型的响应消息：

```
HTTP/1.0200OK
Date:Mon,31Dec200104:25:57GMT
Server:Apache/1.3.14(Unix)
Content-type:text/html
Last-modified:Tue,17Apr200106:46:28GMT
Etag:"a030f020ac7c01:1e9f"
Content-length:39725426
Content-range:bytes554554-40279979/40279980
```

上例第一行表示 HTTP 服务端响应一个 GET 方法。棕色的部分表示响应头域的信息，绿色的部分表示通用头部分，红色的部分表示实体头域的信息。

Location 响应头

Location 响应头用于重定向接收者到一个新 URI 地址。

Server 响应头

Server 响应头包含处理请求的原始服务器的软件信息。此域能包含多个产品标识和注释，产品标识一般按照重要性排序。

实体

请求消息和响应消息都可以包含实体信息，实体信息一般由实体头域和实体组成。实体头域包含关于实体的原信息，实体头包括 Allow、Content- Base、Content-Encoding、Content-Language、 Content-Length、Content-Location、Content-MD5、Content-Range、Content-Type、 Etag、Expires、Last-Modified、extension-header。extension-header 允许客户端定义新的实体 头，但是这些域可能无法未接受方识别。实体可以是一个经过编码的字节流，它的编码方式由 Content-Encoding 或 Content-Type 定义，它的长度由 Content-Length 或 Content-Range 定义。

Content-Type 实体头

Content-Type 实体头用于向接收方指示实体的介质类型，指定 HEAD 方法送到接收方的实体介质类型，或 GET 方法发送的请求介质类型 Content-Range 实体头

Content-Range 实体头用于指定整个实体中的一部分的插入位置，他也指示了整个实体的长度。在服务器向客户返回一个部分响应，它必须描述响应覆盖的范围和整个实体长度。一般格式：

Content-Range:bytes-unitSPfirst-byte-pos-last-byte-pos/entity-legth

例如，传送头 500 个字节次字段的形式：Content-Range:bytes0- 499/1234 如果一个 http 消息包含此节（例如，对范围请求的响应或对一系列范围的重叠请求），Content-Range 表示传送的范围， Content-Length 表示实际传送的字节数。

Last-modified 实体头
应答头
说明
Allow
服务器支持哪些请求方法（如 GET、POST 等）。
Content-Encoding
文 档的编码（Encode）方法。只有在解码之后才可以得到 Content-Type 头指定的内容类型。利用 gzip 压缩文档能够显著地减少 HTML 文档的 下载时间。Java 的 GZIPOutputStream 可以很方便地进行 gzip 压缩，但只有 Unix 上的 Netscape 和 Windows 上的 IE 4、IE 5 才支持它。因此，Servlet 应该通过查看 Accept-Encoding 头（即 request.getHeader("Accept- Encoding")）检查浏览器是否支持 gzip，为支持 gzip 的浏览器返回经 gzip 压缩的 HTML 页面，为其他浏览器返回普通页面。
Content-Length
表 示内容长度。只有当浏览器使用持久 HTTP 连接时才需要这个数据。如果你想要利用持久连接的优势，可以把输出文档写入 ByteArrayOutputStram，完成后查看其大小，然后把该值放入 Content-Length 头，最后通过 byteArrayStream.writeTo(response.getOutputStream()发送内容。
Content-Type
表示后面的文档属于什么 MIME 类型。Servlet 默认为 text/plain，但通常需要显式地指定为 text/html。由于经常要设置 Content-Type，因此 HttpServletResponse 提供了一个专用的方法 setContentTyep。
Date
当前的 GMT 时间。你可以用 setDateHeader 来设置这个头以避免转换时间格式的麻烦。
Expires
应该在什么时候认为文档已经过期，从而不再缓存它？
Last-Modified
文 档的最后改动时间。客户可以通过 If-Modified-Since 请求头提供一个日期，该请求将被视为一个条件 GET，只有改动时间迟于指定时间的文档 才会返回，否则返回一个 304（Not Modified）状态。Last-Modified 也可用 setDateHeader 方法来设置。
Location
表示客户应当到哪里去提取文档。Location 通常不是直接设置的，而是通过 HttpServletResponse 的 sendRedirect 方法，该方法同时设置状态代码为 302。
Refresh
表示浏览器应该在多少时间之后刷新文档，以秒计。除了刷新当前文档之外，你还可以通过 setHeader("Refresh", "5; URL=http://host/path")让浏览器读取指定的页面。
注 意这种功能通常是通过设置 HTML 页面 HEAD 区的 < META HTTP-EQUIV="Refresh" CONTENT="5;URL=http://host/path" > 实现，这是因为，自动刷新或重定向对于那些不能使用 CGI 或 Servlet 的 HTML 编写者十分重要。但是，对于 Servlet 来说，直接设置 Refresh 头更加方便。

注意 Refresh 的意义是“N 秒之后 刷新本页面或访问指定页面”，而不是“每隔 N 秒刷新本页面或访问指定页面”。因此，连续刷新要求每次都发送一个 Refresh 头，而发送 204 状态代码则 可以阻止浏览器继续刷新，不管是使用 Refresh 头还是 < META HTTP-EQUIV="Refresh" ... > 。

注意 Refresh 头不属于 HTTP 1.1 正式规范的一部分，而是一个扩展，但 Netscape 和 IE 都支持它。
Server
服务器名字。Servlet 一般不设置这个值，而是由 Web 服务器自己设置。
Set-Cookie
设置和页面关联的 Cookie。Servlet 不应使用 response.setHeader("Set-Cookie", ...)，而是应使用 HttpServletResponse 提供的专用方法 addCookie。参见下文有关 Cookie 设置的讨论。
WWW-Authenticate

客 户应该在 Authorization 头中提供什么类型的授权信息？在包含 401（Unauthorized）状态行的应答中这个头是必需的。例如， response.setHeader("WWW-Authenticate", "BASIC realm= \ "executives \ "")。
注意 Servlet 一般不进行这方面的处理，而是让 Web 服务器的专门机制来控制受密码保护页面的访问（例如.htaccess）。

Content-type: image/jpeg ---> 返回单张图片

Content-Type: multipart/x-mixed-replace

HTTP 长连接服务器端推技巧.

服务器推送(Server Push)

推送技巧的基础信念是将博览器积极查询消息改为服务器积极发送消息。服务器发送一批数据，博览器揭示这些数据，同时保证与服务器的连接。当服务器必需再次发送一批数据时，博览器揭示数据并坚持连接。尔后，服务器依旧能够发送批量数据，博览器继续揭示数据，顺次类推。

客户端拉曳(Client Pull)

在客户端拖曳技巧中，服务器发送一批数据，在 HTTP 响应或文档头符号中插入号召，让博览器“在 5 秒内再次装入这些数据”或“10 秒内前往某 URL 装入数据”。当指定的工夫到达时，客户端就按照服务器的指示去做，可能刷新目前数据，可能调入新的数据。

其实 push 和 pull 这两种技巧手段极其不同，但目标几乎统一，都是为了给最后用户得体的供给最新消息。

在服务器推送技巧中，HTTP 连接始终坚持着，直到服务器懂得自己已告终发送数据并发送一个告终信号，可能客户端间断连接。而在客户端拖曳技巧中，并不坚持 HTTP 连接，相反，客户端被告诉合时发生新连接，以及发生连接是获得什么数据。

在服务器推送中，独到之处在于“multipart/mixed”款式的 MIME，它能够使一个报文（或 HTTP 响应）包括众多数据项、在客户端拖曳中，独到之处在于 HTTP 响应头标（或等效的 HTML 元素），它能告诉客户端在指定的延随工夫后厉行何种动作。

服务器推送等闲效率要比客户端拖曳效率高，因为它无须为后续数据发生新的连接。由于始终坚持连接，即便未曾数据传输时也是这么，因而服务器定然甘心分配这些 TCP/IP 端口，对于 TCP/IP 端口数有限的服务器这将是一个严重的问题。

客户端拖曳效率低，因为这定然每次为递交数据发生新的连接。然而它无须始终坚持连接。

在切实情形中，发生 HTTP 连接等闲必需花费相当多的工夫，多达一秒甚至更多。因而从功能上琢磨，服务器推送对于最后用户更有吸引力，尤其是对于必需经常更新消息的情形下。

服务器推送相对客户端拖曳的另一点优势是，服务器推送相比拟较轻率扼制。例如，服务器每顺次推送时都坚持一个连接，但它又随时能够关闭其中的任何连接，而 无须要在服务器上设置特异的算法。而客户端拖曳在同样的情形下要繁琐众多，它每次要与服务器发生连接，服务器为打听决将客户端拖曳哀求与特定的最后用户相 称等情形，必需利用相当繁琐的算法。

万一告终服务器推送的 CGI 过程是利用 Shell 脚本语言编写的，有时会存在一些问题。例如，客户端最后用户间断连接，Shell 过程等闲不能当心到，这 将使资源毫无用处的浪费掉，处理这一问题的措施是用 Perl 可能 C 来编写这类 CGI 过程，以利用户间断连接时能够告终运行。

如上所述，在服务器推送中，多个响应中连接始终坚持，使服务器可在任何工夫发送更多的数据。一个显明的利益是服务器全面能够扼制更新数据的工夫和频率。另外，这种措施效率高，因为始终坚持连接。缺点是坚持连接事态会浪费服务器端的资源。服务器推归还比拟轻率间断。

接下来就可能说警?**衿魍扑图记?
服务器在响应哀求时，HTTP 利用 MIME 报文款式来封装数据。等闲一个 HTTP 响应只能包括一个数据块。但 MIME 有一种机制可用一个报文（或 HTTP 响应）表示将多个数据块，这种机制即便成为“multipart/mixed”的规范 MIME 种类。multipart/mixed 报文基本款式如下：
Content-type:multipart/mixed;boundary=ThisRandomString

--ThisRandomString
Content-type:text/plain
第一个对象的数据。
--ThisRandomString
Content-type:text/plain
第二个对象的数据。
--ThisRandomString--

上述报文包括两上数据块，二者的种类都是“text/plain”。最后一个“ThisRandomString”后的两条短线（--）表示报文告终，后?**丛裀謀荨?

对于服务器推送，利用一个“multipart/mixed”种类的变种--multipart/x-mixed-replace。这里，“x-”表示属 于实验种类。“replace”表示每一个新数据块都会轮换前一个数据块。也即便说，新数据不是附带到旧数据尔后，而是轮换它。

下面是切实利用的“multipart/x-mixed-replace”种类：
Content-type:multipart/x-mixed-replace;boundary=ThisRandomString
--ThisRandomString
Content-type:text/plain
第一个对象的数据
--ThisRandomString
Content-type:text/plain
第二个（最后一个）对象的数据。
--ThisRandomString--
利用这一技巧的关键是，服务器并不是推送全副“multipart/x-mixed-replace”报文，而是每次发送后数据块。

HTTP 连接始终坚持，因而服务器能够按自己必需的速度和频率推送新数据，两个数据块之间博览器仅需在目前窗口期待，用户甚至能够到其他窗口做别的事情， 当服务器必需发送新数据时，它只是源（ABC 输入法没那个字*&^$#）传输管道发送数据块，客户端相应的窗口举行自我更新。

在服务器推送技巧中，“multipart/x-mixed-replace”种类的报文由单一的边界限构成，这些边界限瓜分每个数据块。每个数据块都有 自己的头标，因而能够指定对象相干的内容种类和其他消息。由于“multipart/x-mixed-replace”的个性是每一新数据块顶替前一数据 对象，因而博览器中总是揭示最新的数据对象。
“multipart/x-mixed-replace”报文未曾结尾。也即便说，服务器能够永远坚持连接，并发送所需的数据。万一用户不再在博览器窗口 中揭示数据流，可能博览器到服务器间的连接其中（例如用户按“STOP”按钮），服务器的推送才会间断。这是人们利用服务器推送的标兵措施。

当博览器觉察“Content-type”头标或到达头标告终处时，博览器窗口中的前一个文档被打扫，并开始揭示下一个文档。觉察下一个报文边界时，就感受目前数据块（文档）曾经告终。
总之，服务器推送的数据由一组头标（等闲包括“Content-type”）、数据本身和瓜分符（报文边界）三局部构成。博览器看到瓜分符时，它坚持事态不变，直到下一个数据块到达。

将以上观念举行用编程措施告终，就能够获得切实的服务器推送过程。例如，下面的 Unix shell 过程将使博览器每 5 秒揭示顺次服务器上的历程列表：

```
#!/bin/sh
echo "HTTP/1.1 200"
echo "Content-type: multipart/x-mixed-replace;boundary=--ThisRandomString--"
echo ""
echo "--ThisRandomString--"
while true
do
echo "Content-type: text/html"
echo ""
echo "h2Processes on this machine updated every 5 seconds/h2"
echo "time:"
date
echo "p"
echo "plaintext"
ps -el
echo "--ThisRandomString--"
sleep 5
done
```

当心到，边界设置在 sleep 语句之前发送，这能够确保博览器打扫其缓冲区，并揭示所接收到的最新数据。
NCSA HTTPD 用户在内容种类中不能利用空格，包括边界参数。NCSA HTTPD 只能将不带空格字符的字符串作为内容种类。万一在内容种类行中存在空格（冒号后面的空格例外），空格后的任何文本都会被剔除。
下面的示例是准确的：
Content-type: multipart/x-mixed-replace;boundary=ThisRandomString
而下例则不能正常工作，因为它在其中有空格：
Content-type: multipart/x-mixed-replace; boundary=ThisRandomString
服务器推送的另一个优点是它能够针对个体内联图象举行。包括图象的文档能够由服务器定时或定周期举行更新。而告终这一点极其容易：只需使 IMG 元素的 SRC 属性指向推送一系列图象的 URL 即可。

万一服务器推送用于个体内联图象，文档中的图象就会顺次次被新推送来的图象所轮换，而文档本身无须改变（假想文档未曾举行服务器推送）。这么，WEB 版面中有限的动画就可感受静态画面所轮换。

客户端拖曳

客户端拖曳的一个容易用法是使文档按安宁周期积极重载。例如，琢磨下面的 HTML 文档：
<META HTTP-EQUIV="Refresh" CONTENT=1>
<TITLE>Document ONE</TITLE>
<H1>This is Document ONE!</H1>
Here's some text.<P>
万一将它载入扶持动态文档的博览器（Netscape 1.1 以上，Internet Explorer 和 Mosaic 也扶持客户端拖曳），它将每隔一秒将自己重载顺次。
由于 META 元素切实是在 HTML 文档中模仿 HTTP 响应头标，因而它能够告诉博览器将切身消息当作 HTTP 响应利用。上例中的 META 符号相当于：
Refresh:1
这么，切实上即便 HTTP 头标告诉博览器每一秒更新顺次文档。万一必需延时是 12 秒，那么即便这么的号召：

<META HTTP-RQUIV="Refresh" CONTENT=12>
那么它等效于：
Refresh:12

关于客户端的拖曳我也懒的继续写下去，关于怎么使客户端积极申请其他 URL 的数据话，请利用如下：
<META HTTP-EQUIV="Refresh" CONTENT="12;URL=http://icools.yeah.net/">
当心的是，这里的 URL 不能利用相对路径，定然全副指定。

其中工夫间隔能够设置为 0，这么博览器在目前文档揭示告终后，以最快的速度载入新的数据！

建议设置的参数：
HTTP/1.0 200 OK
Connection
Server
Cache-Control
Pragma
Content-Type: multipart/x-mixed-replace;boundary="BOUNDARY"

```
while{
    Content-type
    Content-Length=frame_size+DHT_SIZE
}
```