<http://www.cnblogs.com/9988/archive/2012/03/21/2409086.html>

我用抓包软件抓了http的包，发现accept大多数有两种情况。

第一种：Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, application/x-silverlight, \*/\*  
第二种：Accept: \*/\*5 {: e, A: \_6 ?  h  
这两种形式有什么区别呢？而且这两种形式都有\*/\*，为什么第一种形式还要加入其他的格式呢？

没抓过HTTP的包，你试着把完整包发上来看看，配上当时操作说明。  
不过我认为是下载文件（第二种数据包）和浏览网页（第一种数据包）的不同。

**HTTP协议的头信息详解**

HTTP（HyperTextTransferProtocol）是超文本传输协议的缩写，它用于传送WWW方式的数据，关于HTTP 协议的详细内容请参 考RFC2616。**HTTP协议**采 用了请求/响应模型。客户端向服务器发送一个请求，请求头包含请求的方法、URI、协议版本、以及包含请求修饰符、客户 信息和内容的类似于MIME的消息结构。服务器以一个状态行作为响应，相应的内容包括消息协议的版本，成功或者错误编码加上包含服务器信息、实体元信息以 及可能的实体内容。   
  
　　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。 / W$ \7 ~2 w5 l. r  
  
**通用头域 5 @2 O% \_3 a" A. f\* u**  
　　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。 1 }% d# [5 z8 |  
  
6 U\* k0 ?\* A/ a9 T  
**Cache-Control头域**   
. V  H  {0 A+ G# M\* T7 @4 N) t  
　　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： + F3 Z2 m( e2 z  
  
　　Public指示响应可被任何缓存区缓存。   
$ r" p0 @9 l2 u6 ^6 o  
　　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 / \_6 x+ d\* Z! k9 R9 P# `7 \  
( V9 {6 M) v5 g# Z' l  
　　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。 0 j) U4 q/ i4 ^\* R0 j3 Y  
  
　　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。   h! t+ ]# L$ v  
  
　　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。 ) w% m) f1 M# Y) i& H( m" @# M  
  
　　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。 , z% ?+ h4 {5 b. m8 ~8 H! b& V  
6 X0 ^& x$ R! b3 ]  m  
**请求消息**   
  
　　请求消息的第一行为下面的格式： ' K1 H4 X" ^! a  
  
　　 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。   
6 o2 a4 b7 ~) v4 h: z  
**典型的请求消息：** " u; ~8 \9 V+ f: D; t9 T  
  
　　GET [http://download.microtool.de:80/somedata.exe](http://download.microtool.de/somedata.exe) 1 g$ x( e  \) t, K4 J  
　　Host: download.microtool.de + n1 I5 u7 R$ r; Q6 k3 o  
　　**Accept**:\*/\*   
　　Pragma: no-cache   
　　Cache-Control: no-cache 7 A5 d7 \& y) k! k# u9 A- B  
　　Referer: <http://download.microtool.de/>   
　　User-Agent:Mozilla/4.04[en](Win95;I;Nav)   
　　Range:bytes=554554-   
6 ]! T- m6 K. `7 s  
　　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**' H- K: G( v- ~' u6 p: q, v+ O  
　　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。 3 c% J8 V: u' F; O# c5 t  
9 `- u4 R4 ?8 {6 o& H  
**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。 5 n1 D& S$ f. A! W/ d4 t$ J! l  
  
**Range头域** 4 w: @2 y4 y: J  
  
　　Range头域可以请求实体的一个或者多个子范围。例如， ) e9 x+ \+ w! K9 u9 |& l  
　　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 . |1 a. N% a, m% r, a% `( u9 I1 \_  
　　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   
+ \_4 u  q1 V% [; o" ^7 R  
　　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）。 6 e, b+ e\* i. d+ R  
4 j: m# k) S: T  
**User-Agent头域** " V  i' U7 |6 X- Q  
  
　　User-Agent头域的内容包含发出请求的用户信息。   
# E+ K7 a# M3 H4 O  
**响应消息** + O2 y) G& p- a- O8 J! i/ P/ j4 j  
2 J4 ~. x9 j3 f3 B/ ?. |+ h  
　　响应消息的第一行为下面的格式： + r' j7 v0 z' z' x  
  
　　HTTP-VersionSPStatus-CodeSPReason-PhraseCRLF 3 }9 P! a& B- s" a" `8 j  
  
　　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   
\* V: q1 y8 W/ K5 ?" ^! K% R  
　　2xx:处理成功响应类，表示动作被成功接收、理解和接受 ( Y! v1 \\* H/ ?; A% O  
. o: T$ i$ ^& Q' b) K2 ~" S0 q  
　　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 9 X, l8 O" v; Z5 C  
  
　　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   
' Z  ~0 a( }9 {/ r8 |2 a\* \_# G  
　　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。 % P5 U; x8 t# b: ?) e# F# E  
  
　　典型的响应消息： \* P! A\* ~0 a. z! j5 ?$ a# r  
  
　　HTTP/1.0200OK & L$ Z' ~6 r) q\* \) B& X; G$ ^  
　　Date:Mon,31Dec200104:25:57GMT   
　　Server:Apache/1.3.14(Unix) + a0 s  `7 T: Y4 @7 v  
　　Content-type:text/html   
　　Last-modified:Tue,17Apr200106:46:28GMT 2 F2 }8 w) Z$ B) d# {1 c& G9 r  
　　Etag:"a030f020ac7c01:1e9f"   
　　Content-length:39725426   
　　Content-range:bytes554554-40279979/40279980   
# ?) N& d2 U% D  f0 Q9 r$ N: U  
　　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** + n2 g- n' h- W5 D\* N+ X! a  
; ?\* {5 I5 L! q2 B  
　　Location响应头用于重定向接收者到一个新URI地址。 ( b: \9 L2 B4 J) G" D- l  
/ G& r3 }- D" B0 n" B3 @2 f+ }  
**Server响应头** / {' y7 K  U! d) G: l  
  
　　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。 9 G% a! h/ d6 H5 [  
4 u\* j$ i" M, k" L) q  
**实体** - a! n7 ]  H2 g/ d) p  
  
　　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 " d0 B8 @0 J+ z. W1 G. K3 o7 s+ g  
  
　　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：   
% Q0 c7 J( M' K& z! |! \9 ]; E  
　　Content-Range:bytes-unitSPfirst-byte-pos-last-byte-pos/entity-legth   
" ~8 V) k0 ~- `. Q8 X  
　　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。 / s. d6 V6 b" ]( d  
, I/ m; a; b5 K" v; t  
**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。 , {' W4 @7 T1 [7 P4 P 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。 & y+ M% k) [# e  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" ...＞。  ) A3 R& `\* i0 b, i, E1 s1 I 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）。 |

4 U4 L1 I7 g" a- |% R\* p  
（一）初识HTTP消息头  
但凡搞WEB开发的人都离不开HTTP（超文本传输协议），而要了解HTTP，除了HTML本身以外，还有一部分不可忽视的就是HTTP消息头。+ w$ k/ A: ^\* @\* |! G, ?# c  
做过Socket编程的人都知道，当我们设计一个通信协议时，“消息头/消息体”的分割方式是很常用的，消息头告诉对方这个消息是干什么的，消息体告诉对 方怎么干。HTTP传输的消息也是这样规定的，每一个HTTP包都分为HTTP头和HTTP体两部分，后者是可选的，而前者是必须的。每当我们打开一个网 页，在上面点击右键，选择“查看源文件”，这时看到的HTML代码就是HTTP的消息体，那么消息头又在哪呢？IE浏览器不让我们看到这部分，但我们可以 通过截取数据包等方法看到它。4 U7 L( B, g+ N7 k& s, N& \_  
下面就来看一个简单的例子：  
首先制作一个非常简单的网页，它的内容只有一行：  
<html><body>hello world</body></html>& W7 o0 p2 d' M' x  J- I+ `  
把它放到WEB服务器上，比如IIS，然后用IE浏览器请求这个页面（<http://localhost:8080/simple.htm>），当我们请求这个页面时，浏览器实际做了以下四项工作：  
1 解析我们输入的地址，从中分解出协议名、主机名、端口、对象路径等部分，对于我们的这个地址，解析得到的结果如下：  
协议名：http  
主机名：localhost) J6 q5 ~3 F3 \/ x8 J9 {+ m9 |  
端口：8080# U) m! b$ V7 s+ N" p  
对象路径：/simple.htm\* I+ B0 N. [. F2 P  
2 把以上部分结合本机自己的信息，封装成一个HTTP请求数据包3 S% ?! E, e, n# @" }2 f  
3 使用TCP协议连接到主机的指定端口（localhost, 8080），并发送已封装好的数据包% D% r, i8 n" ~, {1 u6 S  
4 等待服务器返回数据，并解析返回数据，最后显示出来& m, i! S+ u$ g5 v; ]7 F6 F/ e  
由截取到的数据包我们不难发现浏览器生成的HTTP数据包的内容如下：  
GET /simple.htm HTTP/1.1<CR>. c9 Q7 F0 Z2 t) E8 N2 ]1 ]8 h  
**Accept**: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, \*/\*<CR>4 v: q8 d) u" e  I0 |  
**Accept**-Language: zh-cn<CR>: P- U( ^; f\* Y- n8 x  
**Accept**-Encoding: gzip, deflate<CR>9 Y4 U5 ]7 a# M( E  
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322; .NET CLR 2.0.50727)<CR>  
Host: localhost:8080<CR>  
Connection: **Keep**-**Alive**<CR>  
<CR>  
为了显示清楚我把所有的回车的地方都加上了“<CR>”，注意最后还有一个空行加一个回车，这个空行正是HTTP规定的消息头和消息体的分界线，第一个空行以下的内容就是消息体，这个请求数据包是没有消息体的。  
消息的第一行“GET”表示我们所使用的HTTP动作，其他可能的还有“POST”等，GET的消息没有消息体，而POST消息是有消息体的，消息体的内容就是要POST的数据。后面/simple.htm就是我们要请求的对象，之后**HTTP1.1**表示使用的是**HTTP1.1**协议。  
第二行表示我们所用的浏览器能接受的Content-type，三四两行则是语言和编码信息，第五行显示出本机的相关系信息，包括浏览器类型、操作系统信息等，很多网站可以显示出你所使用的浏览器和操作系统版本，就是因为可以从这里获取到这些信息。9 R5 E0 ^, u( o  
第六行表示我们所请求的主机和端口，第七行表示使用**Keep**-**Alive**方式，即数据传递完并不立即关闭连接。  
服务器接收到这样的数据包以后会根据其内容做相应的处理，例如查找有没有“/simple.htm”这个对象，如果有，根据服务器的设置来决定如何处理，如果是HTM，则不需要什么复杂的处理，直接返回其内容即可。但在直接返回之前，还需要加上HTTP消息头。( S" j( j7 F/ H% b  
服务器发回的完整HTTP消息如下：4 V. k% I( x' I5 J% [) ^! ]  
HTTP/1.1 200 OK<CR>  
Server: Microsoft-IIS/5.1<CR>) `9 q5 V$ V2 `\* {  
X-Powered-By: ASP.NET<CR>$ x- K% K0 p! s) o& [9 C% X  
Date: Fri, 03 Mar 2006 06:34:03 GMT<CR>9 H( [( I) u$ `: A$ }% \_  
Content-Type: text/html<CR>" t3 |1 u9 A. J7 `, W$ w- i, M+ r; G  
**Accept**-Ranges: bytes<CR>  I& o, ?( S1 {# C9 i" S2 u  
Last-Modified: Fri, 03 Mar 2006 06:33:18 GMT<CR>7 w0 N) v3 B# N+ n+ ]2 z$ ~  
ETag: "5ca4f75b8c3ec61:9ee"<CR>  
Content-Length: 37<CR>  
<CR>0 f. l7 n, w3 [, \  
<html><body>hello world</body></html>% d; K+ l9 D- v$ D# }  
同样，我用“<CR>”来表示回车。可以看到，这个消息也是用空行切分成消息头和消息体两部分，消息体的部分正是我们前面写好的HTML代码。  
消息头第一行“HTTP/1.1”也是表示所使用的协议，后面的“200 OK”是HTTP返回代码，200就表示操作成功，还有其他常见的如404表示对象未找到，500表示服务器错误，403表示不能浏览目录等等。! }- j& J3 `! [\* d7 D% I  
第二行表示这个服务器使用的WEB服务器软件，这里是IIS 5.1。第三行是ASP.Net的一个附加提示，没什么实际用处。第四行是处理此请求的时间。第五行就是所返回的消息的content-type，浏览器 会根据它来决定如何处理消息体里面的内容，例如这里是text/html，那么浏览器就会启用HTML解析器来处理它，如果是image/jpeg，那么 就会使用JPEG的解码器来处理。9 w5 v" p! s4 S\* z8 i( ~3 ^+ f  
消息头最后一行“Content-Length”表示消息体的长度，从空行以后的内容算起，以字节为单位，浏览器接收到它所指定的字节数的内容以后就会认为这个消息已经被完整接收了。  
/ m4 f( R( X! u: [2 O  
9 A: D1 }6 B1 y5 V  
  
  
理解HTTP消息头 （二）常见的HTTP返回码上一篇文章里我简要的说了说HTTP消息头的格式，注意到在服务器返回的HTTP消息头里有一个“HTTP/1.1 200 OK”，这里的200是HTTP规定的返回代码，表示请求已经被正常处理完成。浏览器通过这个返回代码就可以知道服务器对所发请求的处理情况是什么，每一 种返回代码都有自己的含义。这里列举几种常见的返回码。  
1 403 Access Forbidden如果我们试图请求服务器上一个文件夹，而在WEB服务器上这个文件夹并没有允许对这个文件夹列目录的话，就会返回这个代码。一个完整的403回复可能是这样的：（IIS5.1）  
HTTP/1.1 403 Access Forbidden" s% Q4 a& I4 E% I/ c  
Server: Microsoft-IIS/5.1  
Date: Mon, 06 Mar 2006 08:57:39 GMT% I1 K2 W2 z" N9 e' h2 q1 d: t  
Connection: close  
Content-Type: text/html  
Content-Length: 1726 e" m! f7 S7 B' J1 W: S) t( K  
  
<html><head><title>Directory Listing Denied</title></head>: D2 H3 e$ \_2 ]0 X\* b, \_  K  
<body><h1>Directory Listing Denied</h1>This Virtual Directory does not allow contents to be listed.</body></html>  S% \_' L6 N$ ?  
2 404 Object not found当我们请求的对象在服务器上并不存在时，就会给出这个返回代码，这可能也是最常见的错误代码了。IIS给出的404消息内容很长，除了消息头以 外还有一个完整的说明“为什么会这样”的网页。APACHE服务器的404消息比较简短，如下：\* H. }6 Z1 I: u" @3 X4 |/ m7 T  
HTTP/1.1 404 Not Found  
Date: Mon, 06 Mar 2006 09:03:14 GMT# n" x7 F+ T( ^- d9 S  
Server: Apache/2.0.55 (Unix) PHP/5.0.5  
Content-Length: 291  
**Keep**-**Alive**: timeout=15, max=100- b4 T7 q% E$ Y6 s  
Connection: **Keep**-**Alive**  
Content-Type: text/html; charset=iso-8859-17 M4 |+ ]% S4 w6 h5 \5 B  
4 B, e" t. t$ c' [  
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">  
<html><head>% Y" p7 n" ?& Q  
<title>404 Not Found</title>  
</head><body>  
<h1>Not Found</h1>  
<p>The requested URL /notexist was not found on this server.</p>  
<hr>\* J+ N# e8 C7 z  
<address>Apache/2.0.55 (Unix) PHP/5.0.5 Server at localhost Port 8080</address>( d5 a, i4 m2 k+ W  
</body></html>  
也许你会问，无论 是404还是200，都会在消息体内给出一个说明网页，那么对于客户端来说二者有什么区别呢？一个比较明显的区别在于200是成功请求，浏览器会记录下这 个地址，以便下次再访问时可以自动提示该地址，而404是失败请求，浏览器只会显示出返回的页面内容，并不会记录此地址，要再次访问时还需要输入完整的地 址。  
3 401 Access Denied当WEB服务器不允许匿名访问，而我们又没有提供正确的用户名/密码时，服务器就会给出这个返回代码。在IIS中，设置IIS的安全属性为不允许匿名访问（如下图），此时直接访问的话就会得到以下返回结果：  
: C  i- i0 m6 r0 |/ s: X  
HTTP/1.1 401 Access Denied  
Server: Microsoft-IIS/5.1  
Date: Mon, 06 Mar 2006 09:15:55 GMT  
WWW-Authenticate: Negotiate# v% ~1 }" B+ \_5 v; R% x9 @6 f/ }  
WWW-Authenticate: NTLM; P( U" K3 G0 d/ G  
Connection: close$ a5 L8 M0 b8 d1 B\* T  
Content-Length: 3964) \_- p; d7 ?- U1 u  
Content-Type: text/html  
6 D& N+ b: g# E) U4 [6 j0 n  
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">  
<html dir=ltr>  
……, i$ [1 k- z5 j2 `! \' |/ o; j% Y  
此时浏览器上给出的提示如下图，让我们输入用户名和密码：2 c  a5 R$ z$ W8 X  
3 V" f0 G4 U! P7 h\* e6 I/ T  
因返回信息中消息体较长，只取前面两行内容。注意，如果是用localhost来访问本机的IIS，因IE可以直接取得当前用户的身份，它会和服务器间直接进行协商，所以不会看到401提示。+ W) B0 E2 v) a' c4 q  
当我们在输入了用户名和密码以后，服务器与客户端会再进行两次对话。首先客户端向服务器索取一个公钥，服务器端会返回一个公钥，二者都用BASE64编码，相应的消息如下（编码部分已经做了处理）：  L0 @& D. q" U  h0 m  
GET / HTTP/1.13 i2 B1 S1 v+ {" G8 U# h& B  
**Accept**: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, \*/\*8 b- W. \& A, U+ o& K  
**Accept**-Language: zh-cn  
**Accept**-Encoding: gzip, deflate  
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322; .NET CLR 2.0.50727)  
Host: 192.168.0.55:8080  
Connection: **Keep**-**Alive**' A\* R+ v7 X3 i8 u1 `$ y  
Authorization: Negotiate ABCDEFG……; o\* o0 C& ~+ n1 X( X3 {  
\* \_4 ]7 f% b$ f7 n0 R3 h  
HTTP/1.1 401 Access Denied5 f1 M# S; y; X  
Server: Microsoft-IIS/5.1  
Date: Mon, 06 Mar 2006 09:20:53 GMT  
WWW-Authenticate: Negotiate HIJKLMN……& U0 b' t\* G% I0 T) N  
Content-Length: 3715  
Content-Type: text/html  
  
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">  
<html dir=ltr>\* k  I; B" P' b+ L& e% u  
……  
客户端拿到公钥之后使用公钥对用户名和密码进行加密码，然后把加密以后的结果重新发给服务器：  
GET / HTTP/1.1/ ^( g! r7 E: n' Q% \_2 X) x  
**Accept**: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, \*/\*. ~8 x. ?9 T- f& D( x  
**Accept**-Language: zh-cn  
**Accept**-Encoding: gzip, deflate  
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322; .NET CLR 2.0.50727)  
Host: 192.168.0.55:8080  
Connection: **Keep**-**Alive**  
Authorization: Negotiate OPQRST……\* J& @# t# ?# G- x  
这样，如果验证通过，服务器端就会把请求的内容发送过来了，也就是说禁止匿名访问的网站会经过三次请求才可以看到页面。但因为客户端浏览器已经缓存了公钥，用同一个浏览器窗口再次请求这个网站上的其它页面时就可以直接发送验证信息，从而一次交互就可以完成了。