[TCP/IP协议族(一) HTTP简介、请求方法与响应状态码](http://www.cnblogs.com/ludashi/p/6232060.html)

接下来想系统的回顾一下TCP/IP协议族的相关东西，当然这些东西大部分是在大学的时候学过的，但是那句话，基础的东西还是要不时的回顾回顾的。接下来的几篇博客都是关于TCP/IP协议族的，本篇博客就先简单的聊一下TCP/IP协议族，然后聊一下HTTP协议，然后再聊一下SSL上的HTTP（也就是HTTPS）了。当然TCP/IP协议族是个老生常谈的话题，网络上关于该内容的文章一抓一大把呢，但是鉴于其重要性，还是有必要系统的总结一下的。

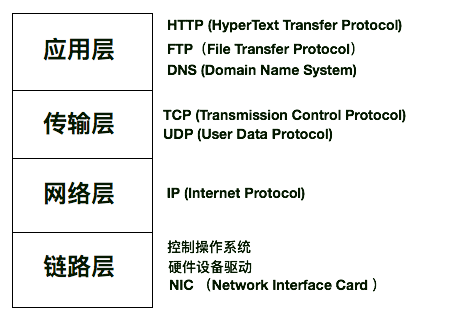
**一、TCP/IP协议组简述**

在聊HTTP与HTTPS之前呢，我们先简单的聊一下TCP/IP协议族。TCP/IP不单单指的就是TCP和IP这两个协议，而是指的与其相关的各种协议。比如HTTP, FTP, DNS, TCP, UDP, IP, SNMP等等都属于TCP/IP协议族的范畴。

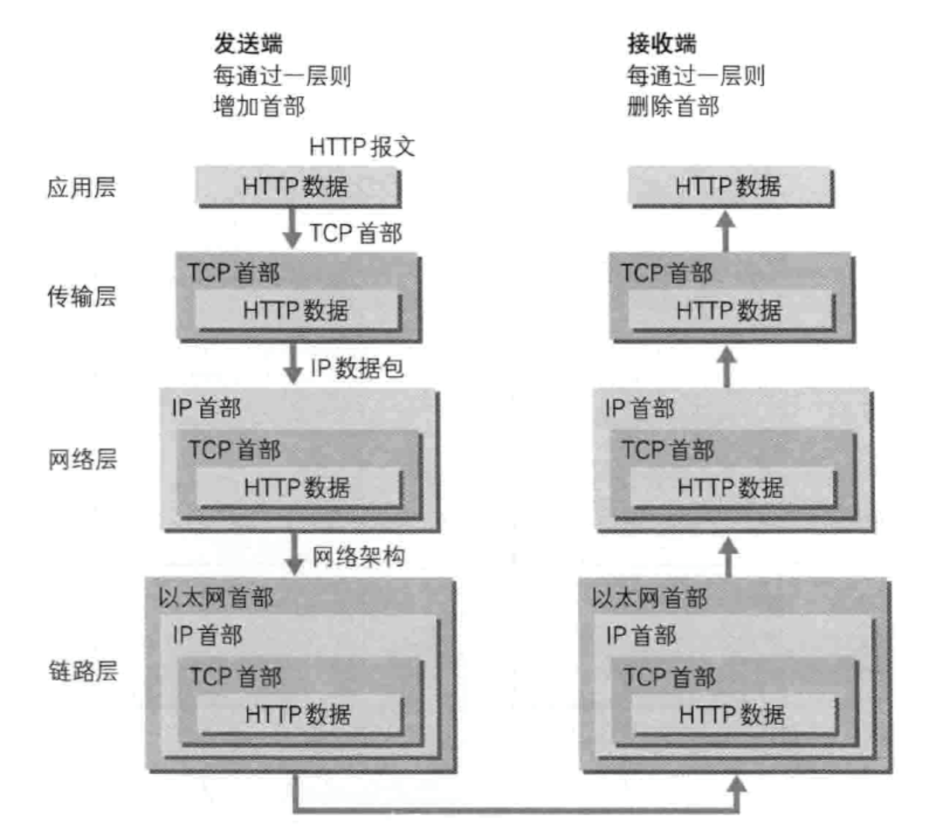
**1.TCP/IP协议的分层**

TCP/IP协议族是分层管理的，在OSI标准中可以分为7层（应用层、表示层、会话层、传输层、网络层、数据链路层、物理层，可记为：应表会传网数物），本篇博客我们采用的是TCP/IP协议族中的四层（应用层、传输层、网络层、链路层）。下方是对四层中每层的简单介绍：

* **应用层**：该层是面向用户的一层，也就是说用户可以直接操作该层，该层决定了向用户提供应用服务时的通信活动。本篇博客要聊的HTTP（HyperText Transfer Protocol：超文本传输协议）就位于该层。我们经常使用的FTP(File Transfer Protocol: 文件传输协议)和DNS (Domain Name System: 域名系统)都位于该层。FTP简单的说就是用来文件传输的。而DNS则负责域名解析的，通过DNS可以将域名（比如：www.cnblogs.com）与IP地址（201.33.xx.09）进行相互的转换。在7层中，又将该层分为：应用层、表示层和会话层。
* **传输层**：应用层的下方是传输层，应用层胡将数据交付给传输层进行传输。TCP(Transmission Control Prococol:传输控制协议)和UDP(User Data Protocol: 用户数据协议)位于该层，当然见名知意，该层是用来提供处于网络连接中的两台计算机直接的数据传输的。TCP建立连接是需要三次握手来确认连接情况，而UDP则没有三次握手的过程。稍后会介绍。
* **网络层**：传输层的下方是网络层，网络层用来处理在网络上流动的数据包，IP(Internet Protocol: 网际协议)就位于这层。该层负责在众多网络线路中选择一条传输线路。当然这个选择传输线路的过程需要IP地址和MAC地址的支持。
* **链路层**：在7层协议中，将链路层分为数据链路层和物理层。该部分主要是用来处理网络的硬件部分，我们常说的NIC（Net Work Card），也就是网卡就位于这一部分，当然光纤也是链路层的一部分。



在TCP/IP协议族中的每次直接在传输数据时的协作关系，以及交互过程，还是引用《图解HTTP》一书上的一张图来解释吧。下图就是这四层协议在数据传输过程中的工作方式。下面这张图还是相当直观的。在发送端是应用层-->链路层这个方向的封包过程，没经过一层都会增加该层的头部。而接收端则是从链路层-->应用层解包的过程，每经过一层则会去掉相应的首部。

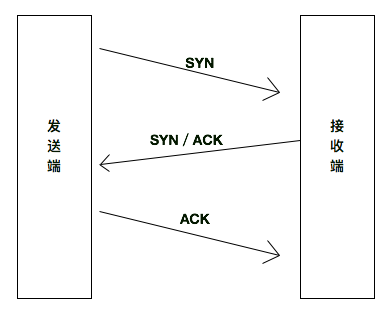


**2、TCP协议的三次握手**

在聊HTTP协议之前，我们先简单的聊一下TCP三次握手的过程，在后面的博客中我们将会对TCP和IP协议进行详述，本篇博客就先简单的聊一下做HTTP协议的基础。

TCP协议位于传输层，为了确保传输的可靠性，TCP协议在建立连接时需要三次握手（Three-way handshaking）。下方这个简图就是TCP协议建立连接时三次握手的过程。

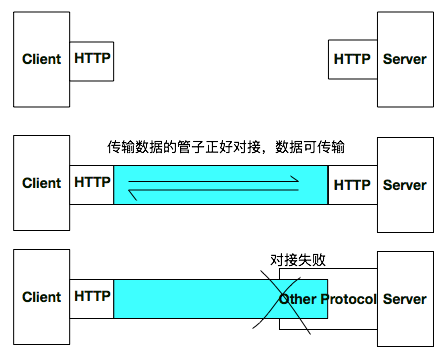
* 第一次握手：发送端发送一个带SYN(Synchronize)标志的数据包给接收端，用于询问接收端是否可以接收。如果可以，就进行第二次握手。
* 第二次握手：接收端回传给发送端一个带有SYN/ACK(Acknowledgement)的数据包，给发送端说，我收到你给我发送的SYN标志了，我再给你传一个ACK标志，你能收到吗？如果发送端收到了SYN/ACK这个数据包，就可以确认接收端收到了之前发送的SYN, 然后进行第三次握手。
* 第三次握手：发送端会给接收端发送一个带有ACK标志的数据包，告诉接收端我可以收到你给我发送的SYN/ACK标志。接收端如果收到了这个来自客户端的ACK标志，就意味着三次握手完成，连接建立，就可以开始传输数据了。



**二、HTTP报文结构**

HTTP协议全称是HyperText Transfer Protocol，即超文本传输协议，用户客户端和服务器之前的通信，目前普遍使用版本为HTTP/1.1。协议本质上就是规范，我们之前提到过的“面向接口”编程，其实就是“面向协议”编程。先定义好类的协议，也就是接口，相关类都遵循该协议，这样一来我们就规范了这些类的调用方式。而HTTP协议是规范客户端和服务器之间通信的协议。也就是说所有的客户端或者服务器都遵循了HTTP这个通信协议，那么也就是意味着他们对外传输数据的接口是一直的，就可以在其中间连接上管道，这样一来就可以进行传输了。

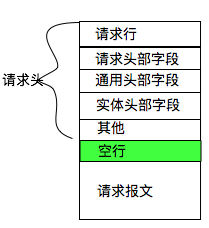
这些协议就是接口，有着共同的通信协议，多个端就可以相互通信。采用相同的协议，就是便于个个设备之间进行沟通交流。HTTP协议的作用如下所示。



HTTP协议的作用是用来规范通信内容的，在HTTP协议中可以分为请求报文和响应报文。顾名思义，请求报文是请求方发出的信息，而响应报文是响应端收到请求后响应的内容。接下来我们就来看看请求报文和响应报文的整体结构。

**1、请求报文（Request Message）结构**

下方是请求报文的整体结构。请求报文主要分为两大部分，一个是请求头（Request Headers）另一个是请求体（Request Body）。这两者之间由空行分割。在请求头中又分为请求行（Request Line），请求头部字段，通用头部字段和实体头部字段等，这个稍后会详细介绍。下方就是请求报文的结构。

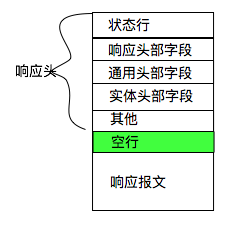


下方这个截图就是请求博客园某个页面时的Request Headers。在请求行中的第一个“GET”是当前请求的方法，稍后会做介绍。中间的就是请求资源的路径，最后一个HTTP/1.1就是当前使用请求协议及其版本。下方这些就是请求头了，稍后会对常用的请求头进行解说。而请求体就是你往服务端传输的数据，比如form表单神马的。

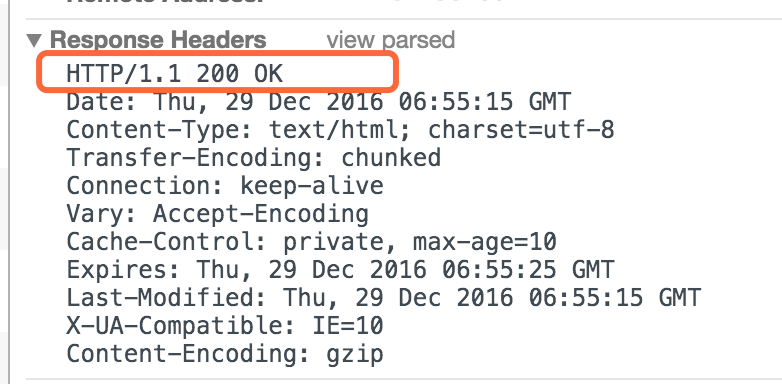


**2、响应报文（Response Message）结构**

聊完请求报文，接下来我们来聊聊响应报文，响应报文的结构与请求报文的结构类似，也分为报文头和报文体。下方就是响应报文的结构图。响应头（Response Headers）分为状态行（State Line），响应头部字段，通用头部字段、实体头部字段等。响应头与响应体中间也是有空行进行分割的。



下方截图就是上述请求报文发出后的响应头，响应体就是对于的HTML等前端资源了。在响应头中，第一行就是状态行，“HTTP/1.1”表示使用的HTTP协议的1.1版本，状态200表示响应成功，"OK"则是状态原因短语。常用状态，稍后会详细介绍。



**三、HTTP的请求方法以及响应状态码**

上面在介绍请求报文中提到的“GET”就是请求请求方法，而在响应报文中提到的“200”状态码，就是稍后要聊的响应状态码。请求方法和响应状态码在HTTP协议中算是比较重要的内容了。之前我们在使用Perfect框架开发服务器端的时候，曾聊过请求方法中的GET、POST、PUT以及DELETE，并且这四种方法可以结合着REST使用。本部分是以HTTP协议的角度来聊的请求方法，所以与之前会有稍稍的不同。本部分我们就来聊一下HTTP协议的请求方法和响应状态码。

**1.请求方法**

接下来我们要聊的请求方法有GET、POST、PUT、HEAD、DELETE、OPTIONS、TRACE、CONNECT。当然上述方法是基于HTTP/1.1的，HTTP/1.0中独有的方法就不说了。

* GET----获取资源
  + GET方法一般用来从服务器上获取资源的方法。服务器端接到GET请求后，就会明白客户端是要从服务器端获取相应的资源，然后就会根据请求报文中相应的参数，将需要的资源返回给客户端。使用GET方式的请求，传输的参数是拼接在URI上的。
* POST----数据提交
  + POST方法一般用于表单提交，将客户端的数据塞到请求体中发送给服务器端。
* PUT----上传文件
  + PUT方法主要用来上传文件，将文件内容塞到请求报文体中，传输给服务器。因为HTTP/1.1的PUT方法自身不带验证机制，所以任何人都可以上传文件，存在安全性，所以上传文件时不推荐使用。但是之前我们在设计接口使用REST标准时，可以使用PUT来做相应内容的更新。
* HEAD----获取响应报文头
  + 响应端收到HEAD请求后，只会返回相应的响应头，不会返回响应体。
* DELETE----删除文件
  + DELETE用于删除URI指定的资源，与PUT一样，自身也是不带验证机制的，不过在REST标准中可以用来做相应API的删除功能。
* OPTIONS----查询支持的方法
  + OPTIONS方法是用来查询服务器可对那些请求方法做出相应，返回内容就是响应端所支持的方法。
* TRACE----追踪路径
  + TRACE方法可追踪请求经过的代理路径，在发送请求时会为Max-Forwards头部字段填入数字，每经过一个代理中转Max-Forwards的值就会减一，直至Max-Forwards为零后，才会返回200。因为该方法易引起XST(Cross-Site Tracing，跨站追踪)攻击，所以不常用呢。
* CONNECT----要求用隧道协议连接代理
  + CONNECT方法要求在与代理服务器通信时建立隧道，实现用隧道协议进行TCP通信。主要使用SSL(Secure Sockets Layer, 安全套接层)和TLS(Transport Layer Security, 传输安全层)协议将通信内容进行加密后经网络隧道传输。

**2、响应状态码**

聊完请求方法后，接下来我们来聊聊HTTP协议的响应状态码。顾名思义，响应状态码是用来标志HTTP响应状态的，响应状态由响应状态码和响应原因短语构成，当然状态码有很多中，本部分就挑出来常用的状态码进行讨论。下方是响应状态码可以分为的几大类：

* 1xx ---- Informational（信息性状态码），表示接受的请求正在处理。
* 2xx ---- Success (成功)，表示请求正常处理完毕。
* 3xx ---- Redirection (重定向)，表示要对请求进行重定向操作，当然其中的304除外。
* 4xx ---- Client Error (客户端错误)，服务器无法处理请求。
* 5xx ---- Server Error (服务器错误)，服务器处理请求时出错。

上面是响应状态码的整体分类，接下来介绍一些常用的响应状态码。

　　(01)、**200 OK** : 表示服务端正确处理了客户端发送过来的请求。

　　(02)、**204 No Content** : 表示服务端正确处理请求，但没有报文实体要返回。

　　(03)、**206 Partial Content** ：表示服务端正确处理了客户端的范围请求，并按照请求范围返回该指定范围内的实体内容。

　　(04)、**301 Moved Permanently**：永久性重定向，若之前的URI保存到了书签，则更新书签中的URI。

　　(05)、**302 Found**：临时重定向，该重定向不会变更书签中的内容。

　　(06)、**303 See Other**：临时重定向，与302功能相同，但是303状态吗明确表示客户端应当采用GET方法获取资源。

　　(07)、**304 Not Modified**: 资源未变更，该状态码与重定向并没有什么关系，当返回该状态码时，告诉客户端请求的资源并没有更新，响应报文体中并不会返回所请求的内容。

　　(08)、**400 Bad Request**： 错误请求，表示请求报文中包含语法错误。

　　(09)、**401 Unauthorized**：请求未认证，表示此发送的请求需要客户端进行HTTP认证（稍后会提到）。

　　(10)、**404 Not Found**：找不到相应的资源，表示服务器找不到客户端请求的资源。

　　(11)、**500 Internal Server Error**：服务器内部错误，表示服务器在处理请求时出现了错误，发生了异常。

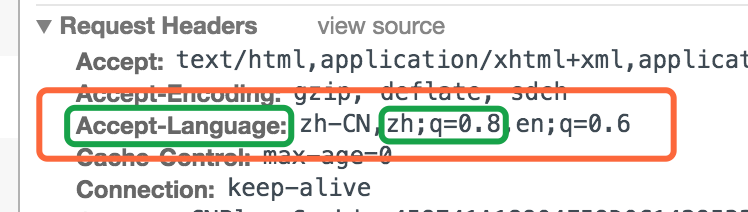
　　(12)、**503 Service Unavailable**：服务不可用，表示服务器处于停机状态，无法处理客户端发来的请求。

[TCP/IP协议族(二) HTTP报文头解析](http://www.cnblogs.com/ludashi/p/6237340.html)

本篇博客我们就来详细的聊一下HTTP协议的常用头部字段，当然我们将其分为请求头和响应头进行阐述。下方是报文头每个字段的格式，首先是头部字段的名称，如Accept，冒号后方紧跟的是该字段名所对应的值，每个值之间有逗号分隔。如果该值需要优先级，那么在值的后方跟上优先级q=0.8(q的值由0~1，优先级从低到高)。值与优先级中间由分号相隔。

头部字段名：值1, 值2;q=0.8

下方就是截取的网络请求中Request Headers的部分内容。红框中的Accept-Language就是头部字段名，冒号后边就是该字段相应的值了。如下所示：



HTTP头部字段可以分为通用头部字段，请求头部字段，响应头部字段以及实体头部字段，下方会给出详细的介绍。

**一.通用头部字段 （General Header Fields）**

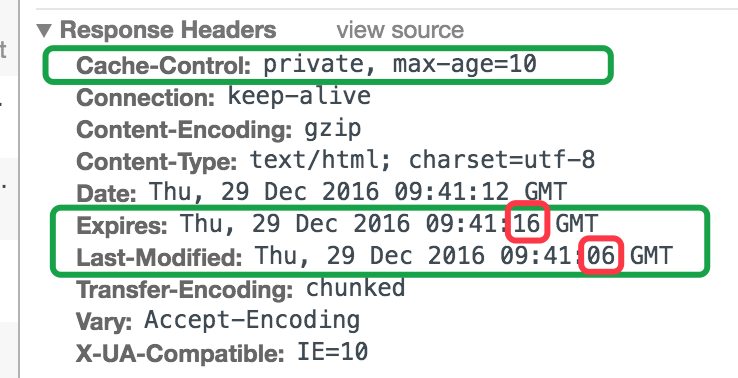
该字段在请求头和响应头都会使用到，下方是常用的通用头部字段：

**1、Cache-Control**

用来操作缓存的工作机制，下方截图响应头中的的Cache-Control的参数为private和max-age=10。private缓存是私有的，仅像特定用户提供相应的缓存信息。如果是public，那么就意味着可向任意方提供相应的缓存信息。max-age = 10表示缓存有效期为10秒。从下方的Expires(过期时间)和Last-Modified(最后修改时间)就可以看出，这两者之间的差值正好是10秒。

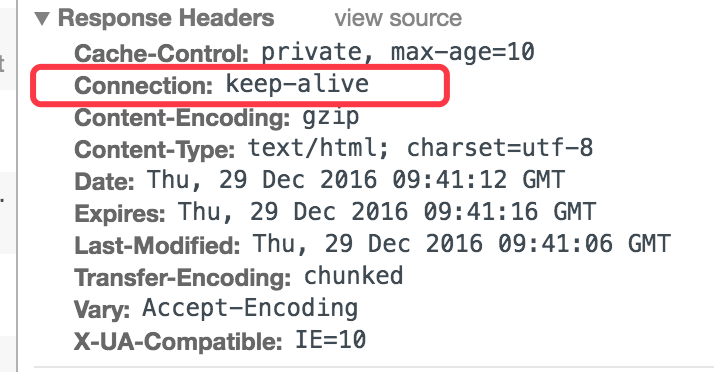
该字段还可以对应其他的参数：

* no-cache：如果是客户端的话，说明客户端不会接收缓存过的响应，要请求最新的内容。而服务器端则表示缓存服务器不能对相应的资源进行缓存。
* no-store：表示缓存不能在本地存储。
* max-age：该参数后方会被赋值上相应的秒数，在请求头中表示如果缓存时间没有超过这个值就返回给我。而在响应头中时，则表示资源在缓存服务器中缓存的最大时间。
* only-if-cached：表示客户端仅仅请求缓存服务器上的内容，如果缓存服务器上没有请求的内容，那么返回504 Gateway Timeout。
* must-revalidata：表示缓存服务器在返回资源是，必须向资源服务器确认其缓存的有效性。
* no-transform：无论请求还是响应，都不能在传输的过程中改变报文体的媒体类型。



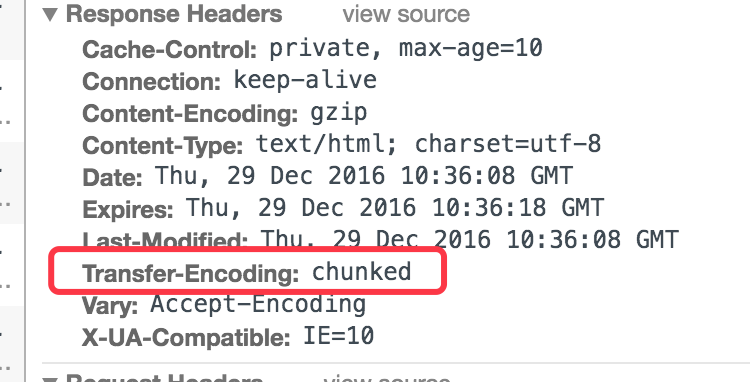
**2、Connection**

该字段可以控制不转发给代理服务器的首部字段以及管理持久连接，下方这个响应报文头中的Connection就是用来管理持久连接的，其参数为keep-alive，就是保持持久连接的意思。可以使用close参数将其关闭。



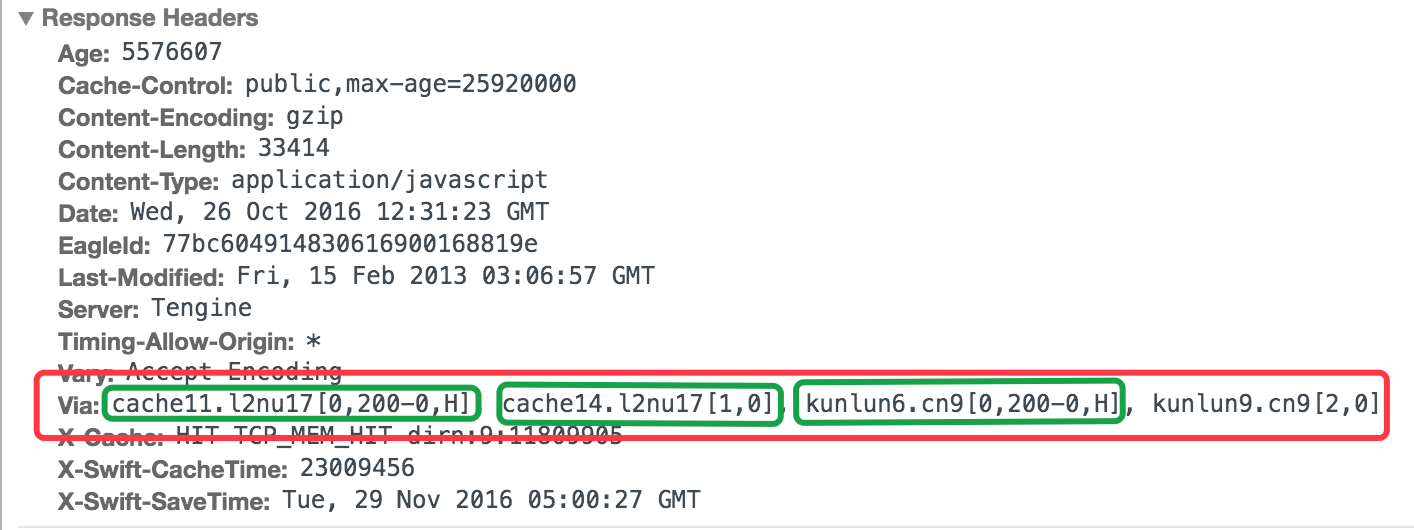
**3、Transfer-Encoding**

该字段表示报文在传输过程中采用的编码方式，在HTTP/1.1的报文传输过程中仅对分块编码有效。下方这个截图就是Transfer-Encoding在Response Header中的使用，后边根的chunked(分块)的参数，说明报文是分块进行传输的。



**4、Via**

该字段是为了追踪请求和响应报文测传输路径，报文经过代理或者网关是会在Via字段添加该服务器的信息，然后再进行转发。



**二.请求头部字段 （Request Header Fields）**

顾名思义，请求头部字段当然是在请求头中才使用的字段。该字段用于补充请求的附加信息，客户端信息等。接下来将给出常用而且比较重要的几个请求头部字段。

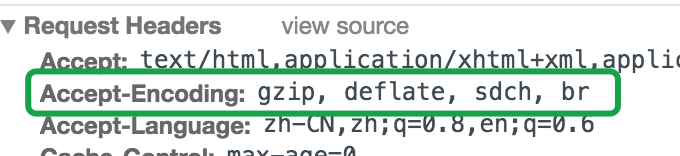
**1  Accept**

该字段可通知服务器用户代理能够处理的媒体类型以及该媒体类型对应的优先级。媒体类型可使用“type/subtype”这种形式来指定，分号后边紧跟着的是该类型的优先级。如下所示。



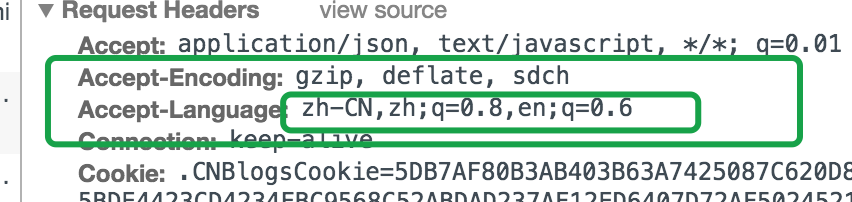
**2  Accept-Encoding**

该字段用来告知服务器，客户端这边可支持的内容编码以及相应内容编码的优先级, 下方就是Accept-Encoding的用法。gzip表示由文件压缩程序gzip(GNU zip)生成的编码格式。compress表示UNIX文件压缩程序compress生成的编码格式。deflate表示组合使用zlib格式以及有deflate压缩算法生成的编码格式。identity表示不执行压缩或者使用一致的默认编码格式。



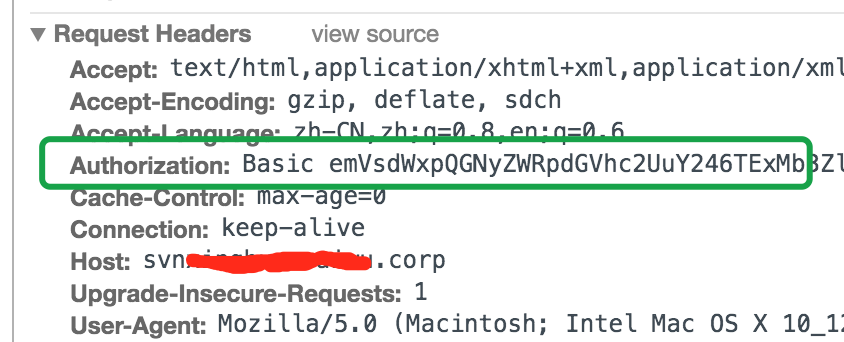
**3 Accept-Language**

该字段用来告知服务器，客户端可处理的自然语言集，以及对应语言集的优先级。以下方的截图为例，Accept-Language后方跟了三个属性，分别是“zh-CN”, "zh;q=0.8"，“en;q=0.6”。也就是说客户端可处理三种自然预言集，zh-CN，其优先级是1（最高）。第二种是zh ，其优先级是0.8，次之。第三个是en，优先级为0.6，优先级在三者之间最低。

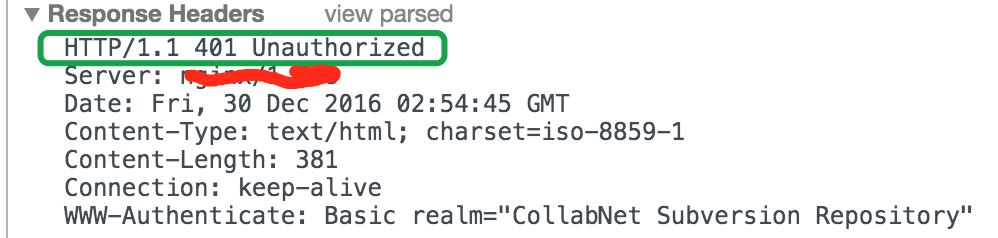


**4 Authorization**

用来告知服务器用户端的认证信息，下方就是连接公司内部SVN系统时需要认证时的请求头部信息。



如果你没有填写认证信息的话，那么就会返回**401 Unauthorized。如下所示：**

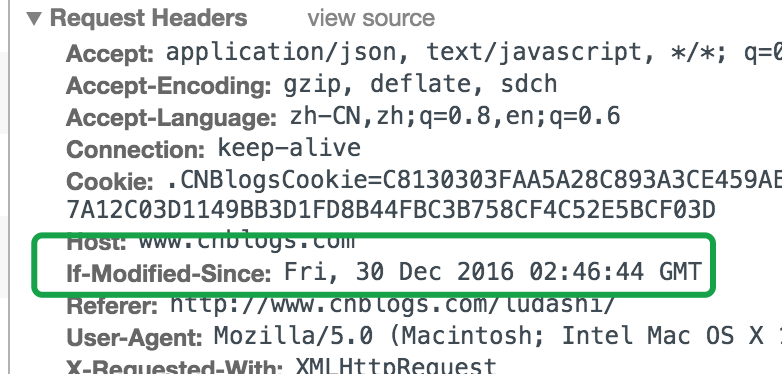


**5 If-Match 与If-None-Match**

上面这两个请求头部字段都是带有逻辑判断的，从上面的英文我们不难看出两者恰好相反。两者后方都跟着串字符串，如If-Match "xcsldjh49773hce", 后边这个字符的匹配对象是ETag(稍后会介绍)。If-Match的请求是如果后方的字符串与ETag相等则服务器端进行请求，否则不进行处理。If-None-Match是If-Match的非操作，同样是匹配ETag, 如果Etag没有匹配成功就处理请求，否则不处理。

**6 If-Modified-Since与If-Unmodified-Since**

If-Modified-Since也是带有逻辑判断的请求头部字段，该字段后方跟的是一个日期，意思是在该日期后发生了资源更新，那么服务器就会处理该请求。If-Unmodified-Since就是 If-Modified-Since的非操作。



**7 If-Range**

if-Range字段后方也是跟的Etag, 该字段要结合着Range字段进行使用。其所代表的意思就是如果Etag匹配成功，请求的内容就按照Range字段所规定的范围进行返回，否则返回全部的内容。用法如下所示：

If-Range: "etag\_code"

Range: bytes=1000-5000

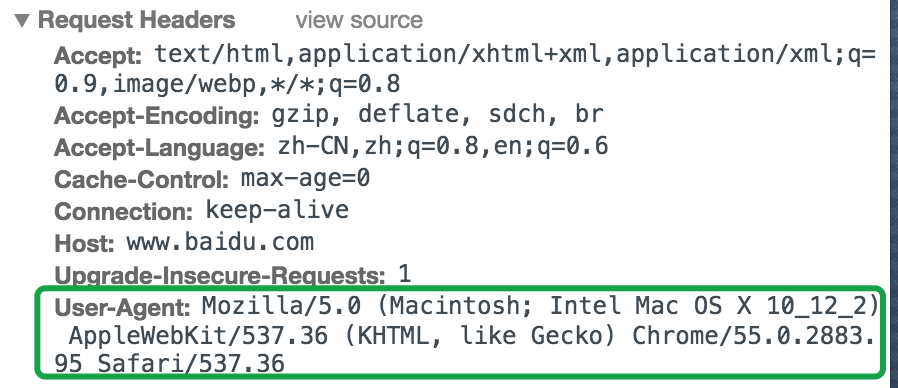
**8  Referer**

 其实**Referer**是一个错误的拼写，但是一直在使用。正确的英文单词应该是Referrer(此处可翻译为:来历、来路)。Referer字段后方跟的是一个URI, 该URI就是发起请求的URI，具体如下所示：



**9  User-Agent**

该字段会将请求方的浏览器和用户代理名称等信息传达给服务器。下方就是从我当前笔记本的Chrome浏览器请求网络时的User-Agent信息。

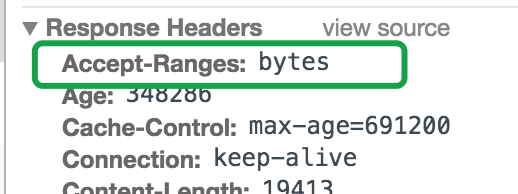
****

**三.响应头部字段 （Request Header Fields）**

聊完请求报文头部字段后，我们接下来来聊一下响应报文头部字段。响应头是由Server向Client返回响应报文中使用的头部信息。用户补充响应的附加信息、服务信息等。下方是几个常见响应头部字段。

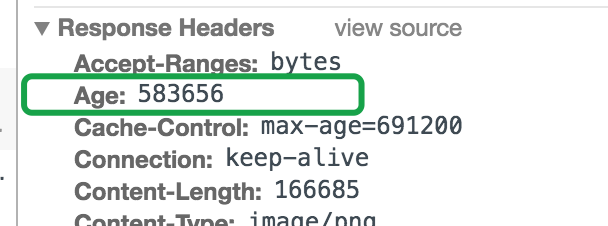
**1  Accept-Ranges**

该字段用来告知客户端服务器那边是否支持范围请求（请求部分内容，请求头中使用Range字段）。Accept-Ranges的值为bytes时，就说明服务器支持范围请求，为none时，说明服务器不支持客户端的范围请求。下方是博客园的页面的加载，从下方可以看出是支持范围请求的，如下所示：



**2 Age**

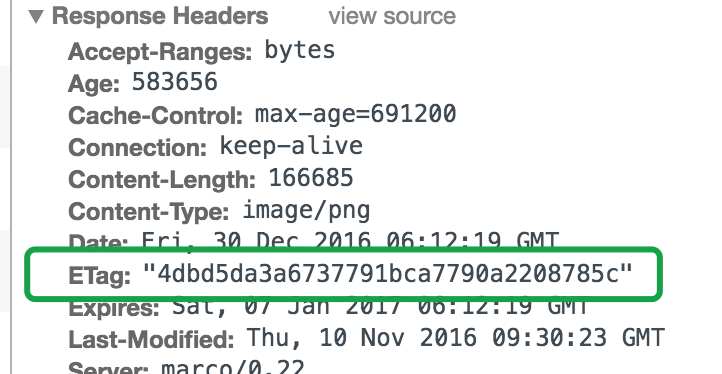
该字段告知客户端，源服务器在多久前创建了该响应。



**3 Etag**

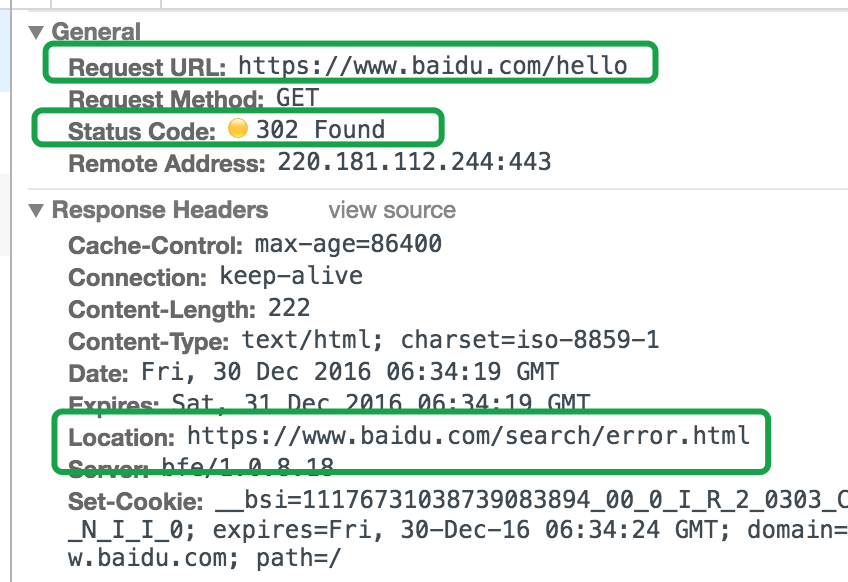
Etag是服务器当前请求的服务器资源（图片，HTML页面等）所对应的一个独有的字符串。不同资源间的Etag是不同的，当资源更新时Etag也会进行更新。

所以结合着请求头中的If-Match等逻辑请求头，可以判断当前Client端已经加载的资源在服务器端是否已经更新了。当初次请求一个资源，如图片时，我们可以将其Etag进行保存，在此请求时，可放在If-None-Match后方，进行资源更新。如果服务器资源并未修改，就不对该请求做出响应。下方就是网页中某张图片对应着的Etag，如下所示。



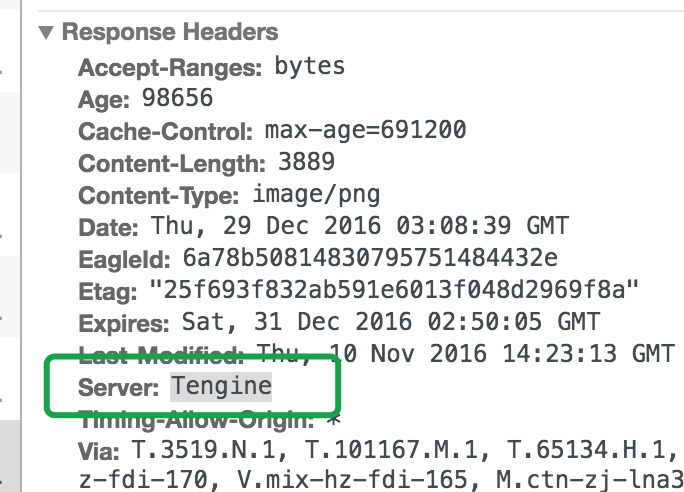
**4 Location**

Location字段一般与重定向结合着使用。下方是我访问“www.baidu.com/hello”这个连接的响应报文。因为服务器上并没有/hello这个资源路径，所以给我重定向了error.html页面，这个重定向的URL就存储在Location字段中，如下所示：



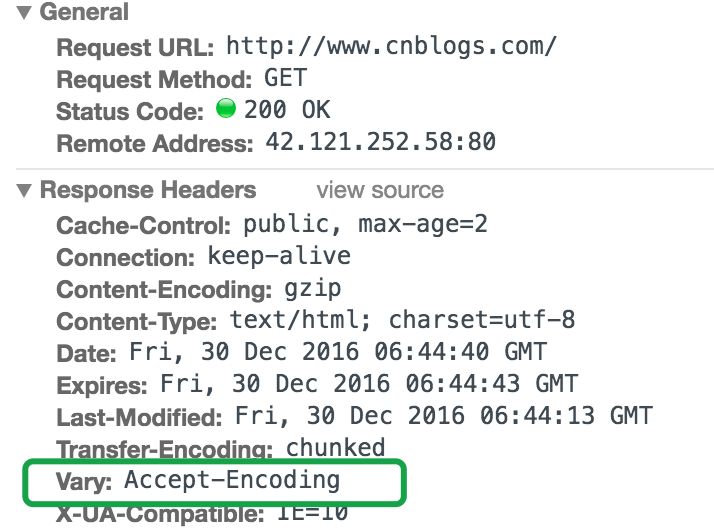
**5 Server**

该响应字段表明了服务器端使用的服务器型号，下方是博客园某张图片的响应头，使用的Web服务器是Tengine, Tengin是淘宝发起的Web服务器项目，是基于Nginx的，关于Tengin的相关内容，请自行Google吧。



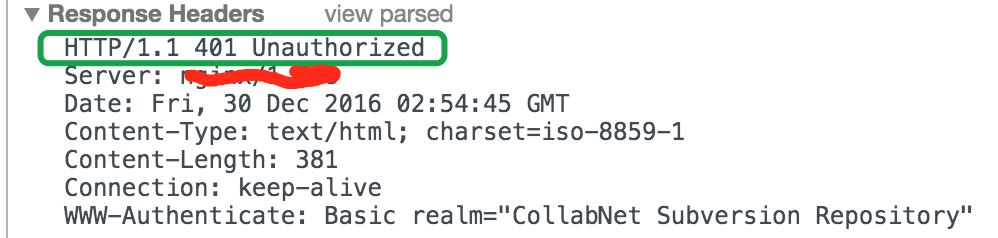
**6 Vary**

Vary可对缓存进行控制，通过该字段，源服务器会向代理服务器传达关于本地缓存使用方法的命令。下方就是Vary的使用，Vary后方的参数是Accept-Encoding。其意思是返回的缓存要以Accept-Encoding为准。当请求的Accept-Encoding的参数与缓存内容的Accept-Encoding参数一致时就返回缓存内容，否则就请求源服务器。



**7 WWW-Authenticate**

该字段用于HTTP的访问认证，在状态码**401 Unauthorized**中肯定带有此字段，该字段用来指定客户端的认证方案（Basic或者Digest）。参数realm的字符串是为了辨别请求URL指定资源所受到的保护策略。如下所示：



**四、实体头部字段（Content Header Fields）**

 接下来我们就来聊聊常见的实体头部字段，实体头部字段是报文实体所使用的头部，用来补充与报文实体相关的信息。

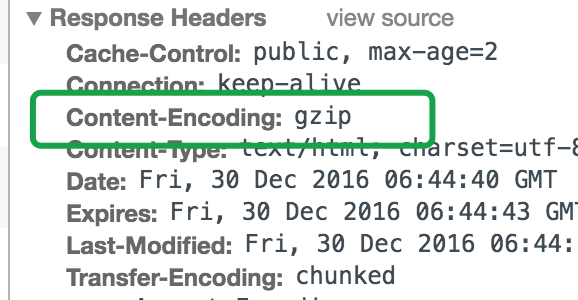
**1  Allow**

该字段用于服务器通知客户端服务器这边所支持的所有请求方法（GET、POST等）。如果服务器找不到客户端请求中所提到的方法的话，就会返回405 Method Not Allowed，于此同时还会把所有能支持的HTTP方法写入到首部字段Allow后返回。

Allow : GET, POST, HEAD, PUT, DELETE

**2 Content-Encoding**

该字段用来说明报文实体的编码方式，下方这段报文头中的Content-Encoding的参数为gzip，说明是使用gzip对报文实体进行压缩的。



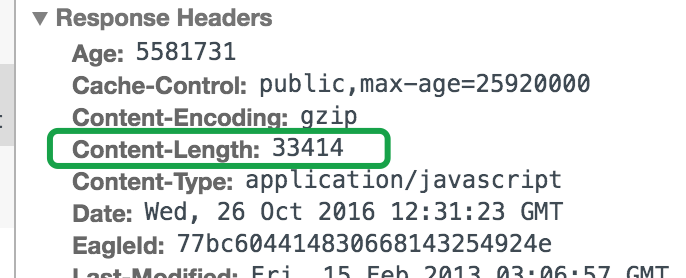
**3 Content-Language**

该字段表示报文实体使用的自然语言，使用方式如下所示：

Content-Language: zh-CN

**4 Content-Length**

顾名思义，该字段用来指定报文实体的字节长度，如下所示：



**5 Content-MD5**

该字段中存储的是报文实体进行MD5加密然后再使用Base64进行编码的字符串。客户端收到响应报文后，可以对报文实体进行MD5加密，然后再对其进行Base64编码，然后与Content-MD5中的字符串进行比较来确定报文是否进行修改，可以说这是一个简单的验签功能。但是此方法并不能确定报文是否被修改了，因为Content-MD5这个值也有可能被篡改。

**五、Cookie相关的头部字段**

因为HTTP协议本身是无状态的，在Web站点中使用Cookie来管理服务器与客户端之间的状态。解析来我就来介绍一下Cookie相关的头部字段。

**1、Set-Cookie**

响应报文中会使用到该字段。当服务器准备开始管理客户端的状态时，会事先告知其各种信息。下方字段是登录知乎时所返回的所要设置的Cookie信息。接下来我们就要对这串Cookie信息进行解析。

* 键值对：在Set-Cookie字段中，“z\_co=Mi4……”这就是要存入Cookie中的信息，当然可以是多个键值对，中间使用逗号进行分割即可。
* Domain：然后是Domain属性，由下方不难看出，Domain中存储的就是Cookie适用对象的域名，若不指定Domain的值，那么默认就是创建Cookie的服务器的域名。
* expire：该字段属性的值是一个时间，也就是Cookie的有效期，若不指定该属性的值，默认就是当前会话有效，关闭浏览器Cookie即失效。
* httponly：设置该属性的目的是让JavaScript脚本无法获取Cookie，其主要目的是防止跨站脚本攻击对Cookie信息的窃取。
* path： 用于限制指定Cookie的发送范围的文件目录。
* Secure：仅在HTTPS安全通信时才会发送Cookie。



**2.Cookie**

请求报文头中会使用该字段，用于将本地存储的Cookie信息发送给服务端。下方就是知乎上每次请求文章所带有的Cookie信息，当然下方只是部分信息，但是我们还是从中可以找到之前我们存储的“z\_co=Mi4……”这个键值对的。



其他比较常见而且比较简单的头部字段就不做过多赘述了，今天博客就先到这儿吧。

作者：[青玉伏案](http://home.cnblogs.com/u/ludashi/)   
出处：<http://www.cnblogs.com/ludashi/>   
本文版权归作者和共博客园共有，欢迎转载，但未经作者同意必须保留此段声明，且在文章页面明显位置给出原文连接，否则保留追究法律责任的权利。   
如果文中有什么错误，欢迎指出。以免更多的人被误导。

分类: [TCP/IP](http://www.cnblogs.com/ludashi/category/931108.html)