# 一个响应头示例与HTTP的五个重要特性

HTTP/1.1 200 OK

Date: Sun, 31 May 2009 17:14:04 GMT

Server: Apache

Last-Modified: Fri, 22 Aug 2008 04:28:16 GMT

ETag: “3075-ddc8d800”

Accept-Ranges: bytes

Content-Length: 12405

Cache-Control: max-age=3153600, public

Expires: Mon, 31 May 2010 17:14:04 GMT

Connection: close

Content-Type: image/jpeg

## 特性一：缓存

网络访问一次花费的代价相对于本地访问来说是及其巨大的。

Cache-Control:max-age 的意义是说“在一周内都不要来烦我”

缓存或者缓存服务器工作的原理就是依据服务器的响应头给出的 Last-Modified ETag Expires Cache-Control 来决定访问原来网页或者访问缓存服务器，或者直接使用本地的资源

**根据Expires和cache-control的内容，浏览器可以确定是否使用缓存或者更新**

**public说明可以从公共的缓存服务器上获取这些资源**

## 特性二：最后修改时间检查

如果**请求方**发送给服务器的内容中使用了If-Modified-Since，如果成立的话，服务器会发送200状态码以及更新的数据给你；否则就会发送304 Modified，告诉你，自己从缓存中取出来用就是了。

## 特性三：ETags

ETags的设计初衷和特性二一样，根据文件内容计算出文件的Hash码，然后如果请求方在请求头中发送了If-None-Match: “ETag”的话，服务器会检查这个文件是更新，相同情况下会返回304 状态码

## 特性四：压缩

发送方在请求头中若使用**Accept-Encoding: identity，**表明其不接受任何形式的压缩算法。

这样在服务器的响应头中可能会发送Vary: Accept-Encoding 来暗示你使用某种形式的压缩算法。如果你发送了gzip/deflate压缩算法，则服务器会在Content-Encoding中提示你它使用的压缩算法

## 特性五：重定向

302 临时重定向 表示这个资源暂时移动到新的url处，不过你下次访问的时候依旧需要访问旧的网址

301 永久重定向 这个资源已经永久移动，你需要在以后的访问中改用新的网址

# 请求头和响应头说明

User-Agent: Python-urllin/3.1

DNT

Expires

Last-Modified

If-Modified-Since

ETag

If-None-Match

Accept-Encoding

Vary

Content-Encoding

Content-Type

Cache-Control · max-age表明最长缓存的时间（秒） （请求方发送no-cache）

Date

Connection 保持通信还是关闭

# 作为http的重要应用：只说明http服务器的使用。

## 理解socket工作的关键是理解socket就像实体的电话，而通信的则是人。显示make socket一个人拿起电话准备bind然后listen，另一个人则拿起电话connect，此时另外一个人就accept了。而且，记住一点，bind类型的socket会获取listen类型的socket来进行读写

连接http的客户端很简单，其实不必是http服务器，任何一个开启了某个服务端口的联网主机都可以被socket编程访问到。

### socket编程思路

TCP服务端：

1 创建套接字，绑定套接字到本地IP与端口

   # socket.socket(socket.AF\_INET,socket.SOCK\_STREAM) , s.bind()

2 开始监听连接                   #s.listen()

3 进入循环，不断接受客户端的连接请求              #s.accept()

4 然后接收传来的数据，并发送给对方数据         #s.recv() , s.sendall()

5 传输完毕后，关闭套接字                     #s.close()

TCP客户端:

1 创建套接字，连接远端地址

       # socket.socket(socket.AF\_INET,socket.SOCK\_STREAM) , s.connect()

2 连接后发送数据和接收数据          # s.sendall(), s.recv()

3 传输完毕后，关闭套接字          #s.close()

服务器端的动作是监听接受和发送，客户端是连接，发送。

send函数在网络busy的情况下可能不会全部发送，最好使用sendall

参考资料：





