**影响Cache的几个HTTP头信息**

注：本篇是关于http头信息cache的一些说明，理论行较强，但是很有用，不管是开发还是运维都得了解，如果还没有研究到这一块的朋友看起来觉得会比较枯燥。

转载于：<http://hi.baidu.com/feilala_fly/blog/item/29b9f200f26c5f03738b6521.html>

Http的Cache机制总共有4个组成部分：

**Cache-Control、Last-Modified（If-Modified-Since）、Etag（If-None-Match） 、Expires**

**服务器响应头**：Last-Modified，Etag  
**浏览器请求头**：If-Modified-Since，If-None-Match

服务器发出Etag，Last-Modified头后，下次浏览器再进行同样的请求，则会发出If-None-Match，If-

Modified-Since头，而后服务器根据这些信息来判断是否需要发送数据，如果没有更新，服务器就简单的

发送一个304状态告诉浏览器用缓存就OK了，不用下载数据了，从而节约了带宽。

**Last-Modified / If-Modified-Since**

Last-Modified是响应头，If-Modified-Since是请求头。Last-Modified把Web组件的最后修改时间告诉客

户端，客户端在下次请求此Web组件的时候，会把上次服务端响应的最后修改时间作为If-Modified-Since

的值发送给服务器，服务器可以通过这个值来判断是否需要重新发送，如果不需要，就简单的发送一个

304状态码，客户端将从缓存里直接读取所需的Web组件。如果有更新，返回HTTP 200和更新的页面内容，

并且携带新的”ETag”和”LastModified”。

    使用这个机制，能够避免重复发送文件给浏览器，不过仍然会产生一个HTTP请求。

**ETag / If-None-Match**

ETag是响应头，If-None-Match是请求头。Last-Modified / If-Modified-Since的主要缺点就是它只能精确到秒的级别，一旦在一秒的时间里出现了多次修改，那么Last-Modified / If-Modified-Since是无法体现的。相比较，ETag / If-None-Match没有使用时间作为判断标准，而是使用一个特征串。Etag把Web组件的特征串告诉客户端，客户端在下次请求此Web组件的时候，会把上次服务端响应的特征串作为If-None-Match的值发送给服务端，服务端可以通过这个值来判断是否需要从重新发送，如果不需要，就简单的发送一个304状态码，客户端将从缓存里直接读取所需的Web组件。因此，HTTP/1.1利用Entity Tag头提供了更加严格的验证。

当服务器发出响应的时候，可以通过两种方式来告诉客户端缓存请求：

**第一种是Expires**，比如：Expires: Sun, 16 Oct 2016 05:43:02 GMT在此日期之前，客户端都会认为缓存是有效的。

不过Expires有缺点，比如说，服务端和客户端的时间设置可能不同，这就会使缓存的失效可能并不能精确的按服务器的预期进行。

**第二种是Cache-Control**，比如：Cache-Control: max-age=3600

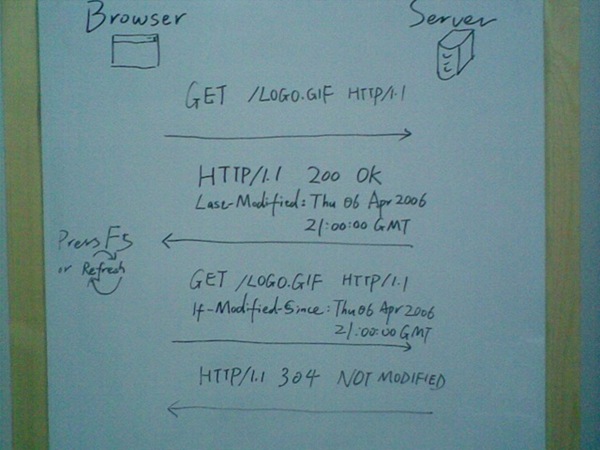
这里声明的是一个相对的秒数，表示从现在起，3600秒内缓存都是有效的，这样就避免了服务端和客户端时间不一致的问题。

但是Cache-Control是HTTP1.1才有的，不适用与HTTP1.0，而Expires既适用于HTTP1.0，也适用于HTTP1.1，所以说在大多数情况下同时发送这两个头会是一个更好的选择，当客户端两种头都能解析的时候，会优先使用Cache-Control基础知识

         1) 什么是”Last-Modified”?   
  
         在浏览器第一次请求某一个URL时，服务器端的返回状态会是200，内容是你请求的资源，同时有一个Last-Modified的属性标记**(Http Reponse Header)**此文件在服务期端最后被修改的时间，格式类似这样：   
  
        Last-Modified: Fri, 12 May 2006 18:53:33 GMT  
  
         客户端第二次请求此URL时，根据 HTTP 协议的规定，浏览器会向服务器传送 If-Modified-Since 报头(**Http Request Header**)，询问该时间之后文件是否有被修改过：   
  
        If-Modified-Since: Fri, 12 May 2006 18:53:33 GMT  
  
         如果服务器端的资源没有变化，则自动返回 **HTTP 304 （Not Changed.）**状态码，内容为空，这样就节省了传输数据量。当服务器端代码发生改变或者重启服务器时，则重新发出资源，返回和第一次请求时类似。从而保证不向客户端重复发出资源，也保证当服务器有变化时，客户端能够得到最新的资源。   
**注**：如果If-Modified-Since的时间比服务器当前时间(当前的请求时间request\_time)还晚，Apache会认为是个非法请求  
  
         2) 什么是”Etag”?   
  
         HTTP 协议规格说明定义ETag为“被请求变量的实体值” （参见 —— 章节 14.19）。 另一种说法是，ETag是一个可以与Web资源关联的记号（token）。典型的Web资源可以一个Web页，但也可能是JSON或XML文档。服务器单独负责判断记号是什么及其含义，并在HTTP响应头中将其传送到客户端，以下是服务器端返回的格式：   
  
        ETag: "50b1c1d4f775c61:df3"  
  
         客户端的查询更新格式是这样的：   
  
        If-None-Match: "50b1c1d4f775c61:df3"  
  
         如果ETag没改变，则返回状态304然后不返回，这也和Last-Modified一样。本人测试**Etag主要在断点下载时比较有用。**          
       Last-Modified和Etags如何帮助提高性能?  
         聪明的开发者会把Last-Modified 和ETags请求的http报头一起使用，这样可利用客户端（例如浏览器）的缓存。因为服务器首先产生 Last-Modified/Etag标记，服务器可在稍后使用它来判断页面是否已经被修改。本质上，客户端通过将该记号传回服务器要求服务器验证其（客户端）缓存。   
         过程如下:  
                 1. 客户端请求一个页面（A）。   
                 2. 服务器返回页面A，并在给A加上一个Last-Modified/ETag。   
                 3. 客户端展现该页面，并将页面连同Last-Modified/ETag一起缓存。   
                 4. 客户再次请求页面A，并将上次请求时服务器返回的Last-Modified/ETag一起传递给服务器。   
                 5. 服务器检查该Last-Modified或ETag，并判断出该页面自上次客户端请求之后还未被修改，直接返回响应304和一个空的响应体。  
  
注：  
1、Last-Modified和Etag头都是由Web Server发出的Http Reponse Header，Web Server应该同时支持这两种头。  
2、Web Server发送完Last-Modified/Etag头给客户端后，客户端会缓存这些头；  
3、客户端再次发起相同页面的请求时，将分别发送与Last-Modified/Etag对应的Http Request Header:If-Modified-Since和If-None-Match。我们可以看到这两个Header的值和Web Server发出的Last-Modified,Etag值完全一样；  
4、通过上述值到服务器端检查，判断文件是否继续缓存；

**关于Etag和Last-Modified网上还有更精辟的解释  
  
1、关于Last-Modified**

HTTP的Response中还会有另外一个Header叫**Last-Modified**，比如  
  
“Last-Modified: Thu, 06 Apr 2006 21:17:12 GMT”，  
  
浏览器访问一个URI得到这样的Resposne之后，就知道这个资源最后一次的修改时间，下次需要再次获得这个资源的时候，会发一个Request给Server，不过这个Request中有一条  
  
“If-Unmodified-Since: Thu, 06 Apr 2006 21:17:12 GMT”，  
  
如果在Server端在这个日期之后对这个资源进行了修改，就会照常返回这个资源给Client端，但是如果没有修改，就会返回一个304 (Not Modified) Response而不返回资源，告诉Client端：“这个资源从上次给你之来从来没改过，你放心用你Cache中的好了。” 一个304 Response比一个静态资源通常小多了，这样就节省了网络带宽。

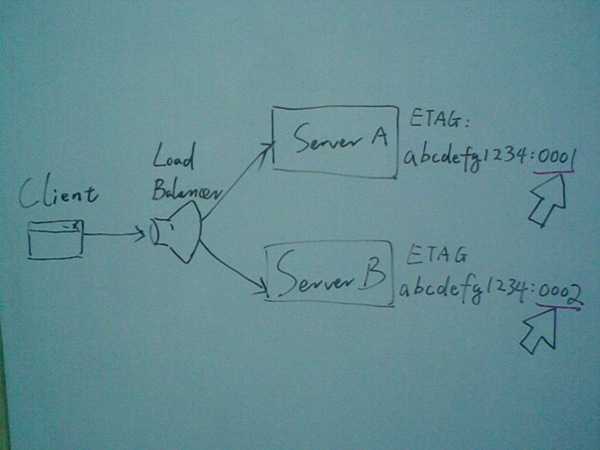


**2、Last-Modified和Expires的区别**

让我们回过头来比较一下Expires和Last-Modified这两个东西，似乎Last-Modified比不上Expires，因为虽然它能够节省一点带宽，但是还是逃不掉发一个HTTP请求出去，而Expires却使得浏览器干脆连HTTP请求都不用发，岂不痛快！那还要Last- Modified这个物体干什么？理想状况的确是这样，不过当用户在IE或者Firefox里面按F5或者点击Refresh按钮的时候（不是在URL栏里重新输入一遍URL然后回车），就算对于有Expires的URI，一样也会发一个HTTP请求出去，所以，Last-Modified还是要用的，而且要和Expires一起用。   
  
**3、Etag**

除了**Last-Modified**，HTTP Response中还可能有另外一个Header: **ETag**，使得Server上的静态资源有点“版本控制”的味道，假如HTTP Response中包含  
  
ETag: "abcdefg1234:0001"  
  
等于告诉Client端，你拿到的这个版本的资源有个ID，叫做abcdefg1234:0001，下次需要发Request索要同一个URI的时候，在Request里面加一条  
  
If-None-Match: "abcdefg1234:0001"  
  
好，Server 端做了一些修改，下次这个Client再来了一个请求，但是这时候资源已经改了，所以返回这个新资源，还有新的tag “ETag: "abcdefg4567:0001"”（这个etag我是胡写的）,这样，Client端等于Cache了两份，在需要索要这个资源的时候，可以包含这样的Header: “If-None-Match: "abcdefg1234:0001" "abcdefg4567:0001"”,这样，即使Server端头脑发热，把这个资源Roll back回原来的版本，依然会返回304 (Not Modified) Response，因为它知道Client端Cache着以前的版本呢，这点功能是Last-Modifed/If-Not-Modified没法做到的。   
  
**4、Etag的弊端**

不过ETag/If-None-Match这点功能实在是个鸡肋，首先，Server端的资源不大可能Roll Back，更重要的是，有可能造成Client Performance下降。对于只有一个Server的网站，没什么问题，但是现在稍微上点规模的网站都需要Scale Out，也就是说需要前端一个Load Balancer，后面接多台Server来处理请求，俗称Cluster，既然是Cluster，那么每个请求到底返回什么结果应该和分配到哪个 Server无关，不过这个ETag可能就坏事了。假如用户的第一次请求分配给Server A，返回“ETag: "abcdefg1234:0001"”，但是第二次请求分配给了Server B，Server B上这个资源和Server A上的一模一样，但是计算出这个资源的ETag是"abcdefg1234:0002"，这下麻烦了，虽然内容一样，但是ETag不匹配，还是浪费了带宽把资源发送了一遍，冤枉啊！而事实上，不同Server上的ETag很有可能不同，对于Apache，ETag的计算考虑了inode，对于 IIS,ETag考虑了metabase的修改版本，要保证不同server上的这些信息一致，有点小难。不过不是有Last-Modified/If- Not-Modified吗？Server端看到If-Modified-Since，对照一下时间对得上，不管If-None-Match，可以直接发回304(Not Modified)呀，很不幸，[RFC2616对这种情况做了规定](http://www.w3.org/Protocols/rfc2616/rfc2616-sec13.html#sec13.3.4)，如果既有If-None-Match又有If-Modified-Since，除非两者不冲突，不然不会返回304。



所以说ETag就是一个害人精，按照Yahoo的建议，别费劲想办法同步不同Server上的ETag了，干脆就把ETag删除得了（缺省，Apache和 IIS都是有ETag的)，我Sniff了一下Yahoo的若干网页返回HTTP Response，的确没有ETag，人家的确是[知行合一](http://morganchengmo.spaces.live.com/blog/cns%219950CE918939932E%211997.entry)。

对于Apache，在httpd.conf或者.htaccess中加一行就搞定了:

**5、Apache中的Etag设置  
补充：**Apache默认开启Etag，可以使用FileEtag来设置  
  
**FileETag none|INode|MTime|Size|All**从apache的实现中http\_etag.c我们可以发现，Apache的Etag包括了Inode|Mtime|Size这些因素。

对于IIS 6，可就有点费劲了，首先，似乎没有办法通过Config来把ETag去掉，查了很多资料，问了很多人，似乎能够去掉ETag的办法只有[写一个ISAPI Filter来弄](http://blogs.msdn.com/yanno/archive/2006/03/03/543176.aspx)，Sniff了一下Microsoft的几个网页的结果显示ETag都稳当当的存在，估计目前真的没有什么好方法。

只好退而取其次，保证不同Server上的ETag一致了。 IIS对Etag的计算算法是ETag = {Filetimestamp:ChangeNumber}, Filetimestamp保持一致没什么问题，ChangeNumber是metabase的change number，就有点难保证Cluster中每个Server都一样了，所以，干脆就把它设成固定值好了，[这个连接](http://support.microsoft.com/kb/922703/)告诉我们该怎么办，很可惜，没有找到彻底删除ETags的配置。

当然转载于此方便以后自己查阅！！！！