浏览器缓存机制

浏览器缓存机制，其实**主要就是HTTP协议定义的缓存机制（如： Expires； Cache-control等）**。但是也有非HTTP协议定义的缓存机制，如使用HTML Meta 标签，Web开发者可以在HTML页面的<head>节点中加入<meta>标签，代码如下：

|  |
| --- |
| **html code** |
| **<META HTTP-EQUIV="Pragma" CONTENT="no-cache">** |

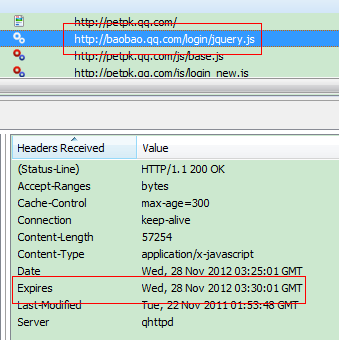
上述代码的作用是告诉浏览器当前页面不被缓存，每次访问都需要去服务器拉取。使用上很简单，但只有部分浏览器可以支持，而且所有缓存代理服务器都不支持，因为代理不解析HTML内容本身。

**下面我主要介绍HTTP协议定义的缓存机制。**

**Expires策略**

Expires是Web服务器响应消息头字段，在响应http请求时告诉**浏览器在过期时间前**浏览器可以直接从浏览器缓存取数据，而无需再次请求。

下面是宝宝PK项目中，浏览器拉取jquery.js web服务器的响应头：

[](http://images.cnblogs.com/cnblogs_com/skynet/201211/201211281402412240.png)

***注：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,31 Dec 2001 04:25:57GMT。***

Web服务器告诉浏览器在2012-11-28 03:30:01这个时间点之前，可以使用缓存文件。发送请求的时间是2012-11-28 03:25:01，即缓存5分钟。

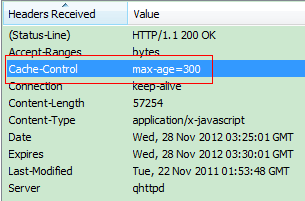
不过Expires 是HTTP 1.0的东西，现在默认浏览器均默认使用HTTP 1.1，所以它的作用基本忽略。

**Cache-control策略（重点关注）**

Cache-Control与Expires的作用一致，都是指明当前资源的**有效期**，控制浏览器是否直接从浏览器缓存取数据还是重新发请求到服务器取数据。只不过Cache-Control的**选择更多，设置更细致**，如果同时设置的话，其**优先级高于Expires**。

|  |
| --- |
| **http协议头Cache-Control    ：** |
| **值可以是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：**   1. **Public指示响应可被任何缓存区缓存。** 2. **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。** 3. **no-cache指示请求或响应消息不能缓存** 4. **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。** 5. **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。** 6. **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。** 7. **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。** |

还是上面那个请求，web服务器返回的Cache-Control头的值为max-age=300，即5分钟（和上面的Expires时间一致，这个不是必须的）。

[](http://images.cnblogs.com/cnblogs_com/skynet/201211/201211281402423419.png)

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

Last-Modified/If-Modified-Since要配合Cache-Control使用。

  Last-Modified：标示这个响应资源的最后修改时间。web服务器在响应请求时，告诉浏览器资源的最后修改时间。

  If-Modified-Since：当资源过期时（使用Cache-Control标识的max-age），发现资源具有Last-Modified声明，则再次向web服务器请求时带上头 **If-Modified-Since**，表示请求时间。**web服务器收到请求后发现有头If-Modified-Since 则与被请求资源的最后修改时间进行比对**。若最后修改时间较新，说明资源又被改动过，则响应整片资源内容（写在响应消息包体内），HTTP 200；若最后修改时间较旧，说明资源无新修改，则响应HTTP 304 (无需包体，节省浏览)，告知浏览器继续使用所保存的cache。

**Etag/If-None-Match**

Etag/If-None-Match也要配合Cache-Control使用。

  Etag：web服务器响应请求时，告诉浏览器当前资源在服务器的唯一标识（生成规则由服务器觉得）。*Apache中，ETag的值，默认是对文件的索引节（INode），大小（Size）和最后修改时间（MTime）进行Hash后得到的*。

  If-None-Match：当资源过期时（使用Cache-Control标识的max-age），发现资源具有Etage声明，则再次向web服务器请求时带上头If-None-Match**（Etag的值）**。**web服务器收到请求后发现有头If-None-Match 则与被请求资源的相应校验串进行比对，决定返回200或304**。

**既生Last-Modified何生Etag？**

你可能会觉得使用Last-Modified已经足以让浏览器知道本地的缓存副本是否足够新，为什么还需要Etag（实体标识）呢？HTTP1.1中Etag的出现主要是为了解决几个Last-Modified比较难解决的问题：

  Last-Modified标注的最后修改只能精确到秒级，如果某些文件在1秒钟以内，被修改多次的话，它将不能准确标注文件的修改时间

  如果某些文件会被定期生成，当有时内容并没有任何变化，但Last-Modified却改变了，导致文件没法使用缓存

  有可能存在服务器没有准确获取文件修改时间，或者与代理服务器时间不一致等情形

Etag是服务器自动生成或者由开发者生成的对应资源在服务器端的唯一标识符，能够更加准确的控制缓存。**Last-Modified与ETag是可以一起使用的，服务器会优先验证ETag，一致的情况下，才会继续比对Last-Modified，最后才决定是否返回304**。

**用户行为与缓存**

浏览器缓存行为还有用户的行为有关！！！

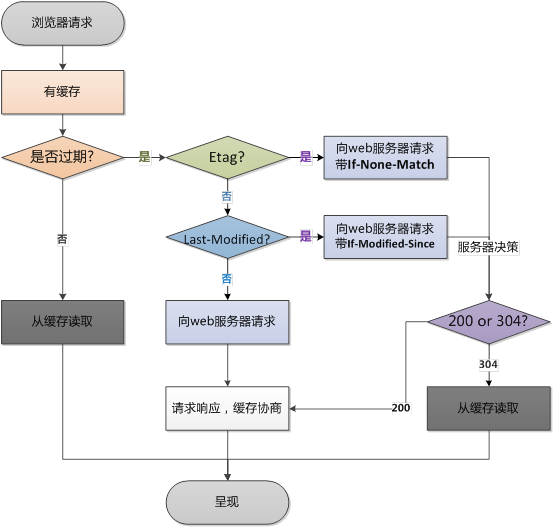
|  |  |  |
| --- | --- | --- |
| **用户操作** | **Expires/Cache-Control** | **Last-Modified/Etag** |
| **地址栏回车** | 有效 | 有效 |
| **页面链接跳转** | 有效 | 有效 |
| **新开窗口** | 有效 | 有效 |
| **前进、后退** | 有效 | 有效 |
| **F5刷新** | 无效 | 有效 |
| **Ctrl+F5刷新** | 无效 | 无效 |

**总结**

浏览器第一次请求：

[](http://images.cnblogs.com/cnblogs_com/skynet/201211/201211281402437389.png)

浏览器再次请求时：

[](http://images.cnblogs.com/cnblogs_com/skynet/201211/201211281402438011.png)