

前端cache 策略

主讲人 郭锋棉 coverguo

什么是cache

浏览器缓存 (Browser Caching) 是为了加速浏览, 浏览器在用户磁盘上对最近请求过的文档进行存储, 当访问者再次请求这个页面时, 浏览器就可以从本地磁盘显示文档, 这样就可以加速页面的阅览

Cache的作用

有两个主要的理由让人们使用缓存:

- 1、减少延迟 — 因为所发出的网页请求是指向更接近客户端的缓存而不再是源服务器端, 因此请求所花费时间更短, 提高用户体验。
- 2、降低网络负荷 — 避免网络拥塞, 减少请求量, 减少输出带宽.

Cache相关header属性:

在HTTP请求的response返回header中, 涉及缓存属性 — Expires 和 Cache-Control 和 Etag。

HTTP/1.0规范, Expires设置对象的有效期;

HTTP 1.1 引入了 Cache-Control 响应头参数以给站长们更多控制网站内容的权力, 同时弥补了 Expires的局限.

Cache-Control 的参数包括:

max-age =[单位: 秒 seconds] — 设置缓存最大的有效时间. 类似于 Expires, 但是这个参数定义的是时间大小 (比如: 60) 而不是确定的时间点.单位是[秒 seconds].

public — 响应会被缓存, 并且在多用户间共享。正常情况, 如果要求 HTTP 认证,响应会自动设置为 private.

private — 响应只能够作为私有的缓存(e.g., 在一个浏览器中), 不能再用户间共享。

no-cache — 响应不会被缓存,而是实时向服务器端请求资源。这一点很有用, 这对保证 HTTP 认证能够严格地禁止缓存以保证安全性很有用 (这是指页面与public结合使用的情况下) .既没有牺牲缓存的效率, 又能保证安全。

no-store — 在任何条件下, 响应都不会被缓存, 并且不会被写入到客户端的磁盘里, 这也是基于安全考虑的某些敏感响应才会使用这个。

HTTP 1.1 协议同时引入一个新的验证器，名为 **Etag**。这种验证器拥有特殊的标示符，它来自于服务器，并且随着响应资源的改变而改变。简单点即 服务器响应时给请求URL标记，并在HTTP响应头中将其传送到客户端，在Http响应头中包含

Etag:“5d8c72a5edda8d6a:3239”标识，等于告诉客户端，你拿到的这个的资源有表示 ID：5d8c72a5edda8d6a:3239。当下次需要发Request索要同一个URI的时候，浏览器同时发出一个If-None-Match 报头(Http RequestHeader)此时包头中信息包含上次访问得到的，即 Etag:“5d8c72a5edda8d6a:3239”标识：

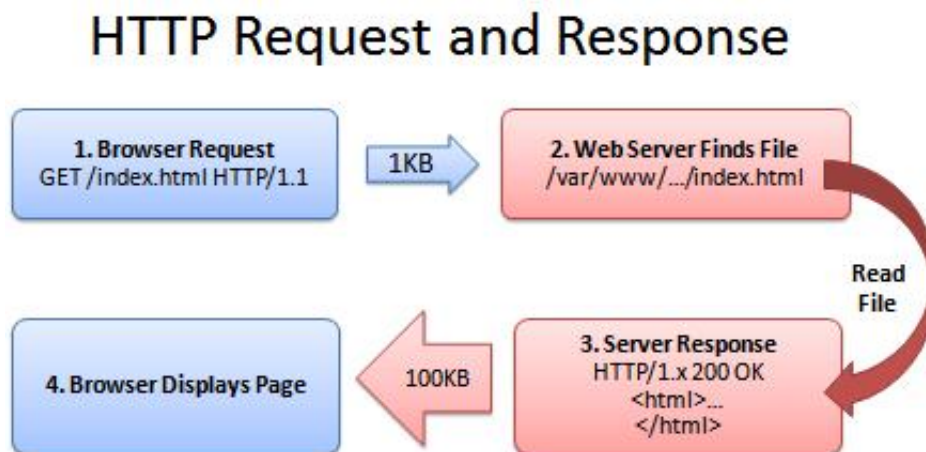
If-None-Match:“5d8c72a5edda8d6a:3239”

这样，客户端等于Cache了两份，服务器端就会比对2者的Etag。如果If-None-Match为False，返回304(缓存读取)，不返回200（服务器读取）。

几乎所有的缓存都是使用了 Last-Modified作为验证器，而Etag有更高的优先权

HTTP Caching 用好了，可以极大的减小服务器负载和减少网络带宽。十分有必要深入了解下 http 的 caching 协议。

先来看下请求/响应过程：

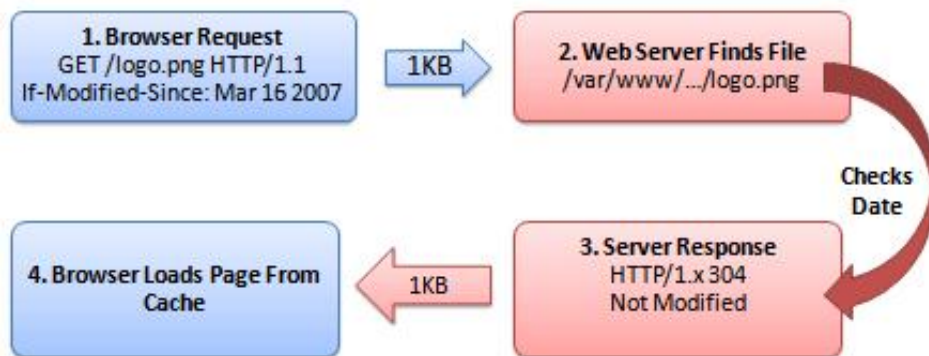


Cache相关策略原理：

1、用 Last-Modified 头

在第一次请求的响应头返回 Last-Modified 内容，时间格式如：Wed, 22 Jul 2009 07:08:07 GMT。是零时区的 GMT 时间，servlet 中可以用 response.addDateHeader (“Last-Modified”, date.getTime ()); 加入响应头。

HTTP Cache: Last-Modified



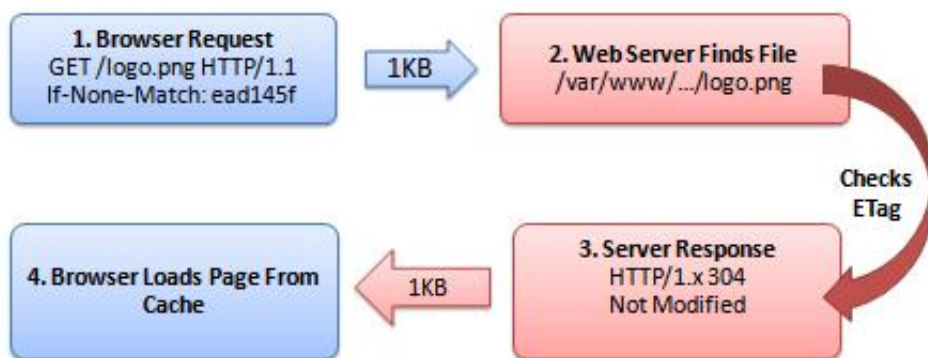
Last-Modified 与 If-Modified-Since 对应的，前者是响应头，后者是请求头。服务器要处理 If-Modified-Since 请求头与 Last-Modified 对比看是否有更新，如果服务器端的资源没有变化，则自动返回HTTP304 (NotChanged.) 状态码，内容为空，这样就节省了传输数据量。当服务器端代码发生改变或者重启服务器时，则重新发出资源，返回和第一次请求时类似。从而保证不向客户端重复发出资源，也保证当服务器有变化时，客户端能够得到最新的资源。

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，会认为是个非法请求

2、用 Etag 头

很多时间可能不能用时间来确定内容是否有更新。那可以用 Etag 头，etag 是以内容计算一个标识。计算的方式可以自己决定，比如可以用 crc32、md5等。

HTTP Cache: If-None-Match



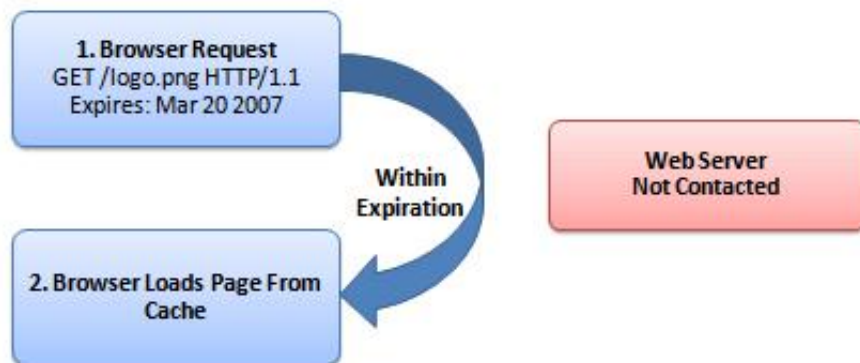
Etag 和 If-None-Match

Etag 与 If-None-Match 是对应的，前者是响应头，后者是请求头。服务器要判断请求内容计算得到的 etag 是否与请求头 If-None-Match 是否一致，如果一致就表示没有更新，返回 304 就可，否则按正常请求处理。

3、用 Expires 头，过期时间

当请求的内容有 Expires 头的时候，浏览器会在这个时间内不去下载这个请求的内容（这个行为对 F5无效，用 IE7，Firefox 3.5 试了，有效的比如：在地址输入后回车）。

HTTP Cache: Expires



在 servlet 中可以用 `response.addDateHeader("Expires", date.getTime());` 添加过期内容。

ps：在 fiddler 中可以看到 请求的结果为 (form Cached) 状态的。

4、用 Cache-Control 的 max-age

max-age 的值表示，多少秒后失效，在失效之前，浏览器不会去下载请求的内容（当然，这个行为对 F5无效）。比如：服务器写 max-age 响应：`response.addHeader("Cache-Control", "max-age=10");`

ps：如果你还要加一些 Cache-Control 的内容，比如：private，最好不要写两个 addHeader，而是一个 `response.addHeader("Cache-Control", "private, max-age=10");`；否则 ie 可能对 max-age 无效，原因它只读第一个 Cache-Control 头。

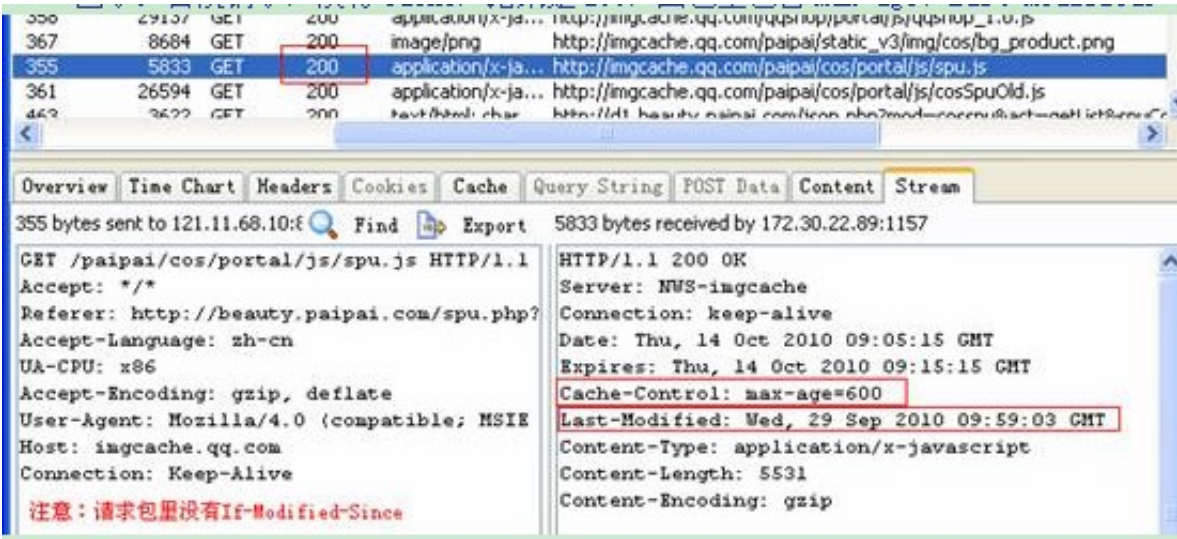
5、max-age 和 Last-Modified (If-Modified-Since) 的组合

我们的业务中用的比较多的是 max-age 和 Last-Modified (If-Modified-Since) 的组合，现在详细介绍一下，这个策略是在给出 max-age 的同时，给出一个资源的验证方式：Last-Modified，标示这个响应资源的最后修改时间，例如 Last-Modified: Thu, 08 Apr 2010 15:01:08 GMT，这个属性只有配合 Cache-control 的时候才有实际价值，利用资源的可校验性，可以实现在 cache 的资源超过 max-age 浏览器再次请求时的 304 响应，令浏览器再次使用之前的 cache

第一步：

- 1、浏览器第一次请求资源 `http://xxx/a.js`
- 2、浏览器查询临时文件目录发现无 cache 内容，于是发出请求到 web server
- 3、Web server 接到请求，响应资源，并设定 `Cache-control: max-age=600`, `Last-Modified: Wed, 20 Jan 2014 09:59:03 GMT`
- 4、浏览器接到响应，将内容展示的同时，在临时文件目录以“`http://xxx/a.js`”为 key 缓存这个响应的内容

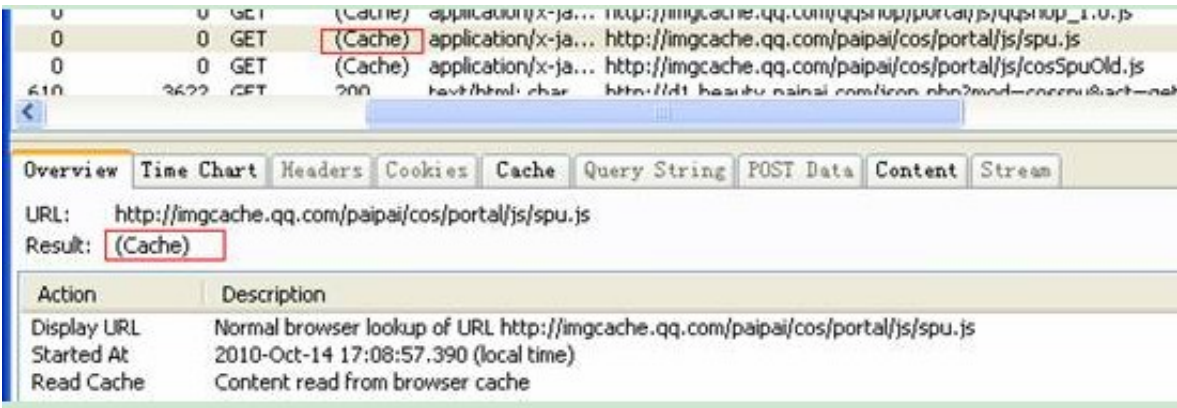
图示：首次请求，没有 cache，结果是 200，回包里包含 max-age、Last-Modified。



第二步:

- 1、离第一步请求间隔时间不到10分钟
- 2、浏览器再次请求资源`http://xxx/a.js`
- 3、浏览器查询临时文件目录发现有cache内容，于是检查`max-age`，还未过期，直接读取，响应给用户。（HTTP状态(Cache)）

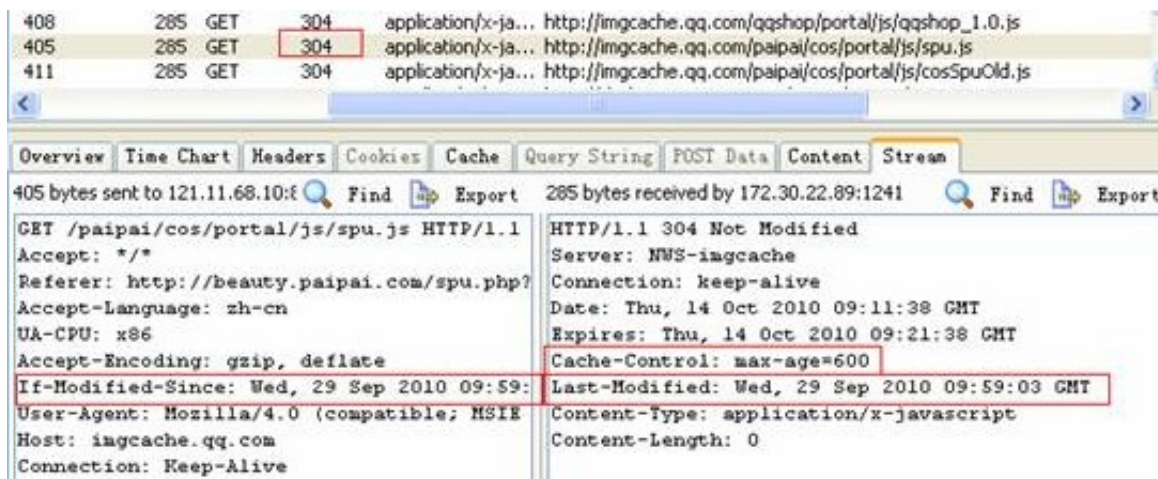
图示：在不到10分钟的间隔里，再次请求这个js文件，直接从cache里读取了内容



第三步:

- 1、离第一步请求间隔时间已超过10分钟
- 2、浏览器再次请求资源`http://xxx/a.js`
- 3、浏览器查询临时文件目录发现有cache内容，于是检查`max-age`，已经过期，发现资源带有 `Last-Modified`，于是在请求包中带上 `If-Modified-Since: Wed, 20 Jan 2014 09:59:03 GMT`，发请求给web server
- 4、Web server收到请求后发现有 `If-Modified-Since`，于是和被请求资源的最后修改时间进行比对,如果修改时间比请求包的时间新，说明资源已经被改过，则带包体响应回复整个资源内容（HTTP状态200）；如果修改时间不比请求包的时间新，说明资源在这段时间内都没改过，无需回复整个资源，则仅响应包头，不带包体（HTTP状态304），告诉浏览器继续使用临时目录里的cache内容展示。

图示：不做清除cache的操作，1个小时以后，再次请求这个js文件，`max-age`已经过期了，但修改时间没变，所以304。



6、max-age和ETag的组合

这个策略是，在给出max-age的同时，给出另一种资源的验证方式：ETag，标示这个响应资源由开发者自己确定的验证标识，例如 ETag: “12345678”

同样这个属性也只有配合Cache-control的时候才有实际价值，是声明校验资源的方式，ETag的使用为实现304响应提供了更多的灵活性

第一步：

- 1、浏览器第一次请求资源http://xxx/b.css
- 2、浏览器查询临时文件目录发现无cache内容，于是发出请求到web server
- 3、Web server接到请求，响应资源，并设定Cache-control:max-age=3600, ETag: "12345678"
- 4、浏览器接到响应，将内容展示的同时，在临时文件目录 以“http://xxx/b.css”为key缓存这个响应的内容

第二步：

- 1、离第一步请求间隔时间不到60分钟
- 2、浏览器再次请求资源http://imgcache.qq.com/qgshow_v3/css/index_.css
- 3、浏览器查询临时文件目录发现有cache内容，于是检查max-age，还未过期，直接读取，响应给用户。（HTTP状态“(Cache)”）

第三步：

- 1、离第一步请求间隔时间已超过60分钟
- 2、浏览器再次请求资源http://xxx/b.css
- 3、浏览器查询临时文件目录发现有cache内容，于是检查max-age，已经过期，发现资源带有ETag，于是在请求包中带上If-None-Match: "12345678"，发请求给web server
- 4、Web server收到请求后发现有If-None-Match，于是和被请求资源的验证串进行比对，如果校验串的内容不一致，则返回整个资源包体（HTTP状态200），如果校验串的内容一致，仅返回包头（HTTP状态304），告诉浏览器继续使用临时目录里的cache内容展示。

Cache相关头属性区别：

Last-Modified 与 Etag 头（即是方式 1 和 2）还是要请求服务器的，只是仅返回 304 头，不返回内容。所以浏览怎么 F5，304 都是有效的。但用 Ctrl+F5 是全新请求的（这是浏览器行为，不发送缓存相关的头）。

Expires 头与 max-age 缓存是不需要请求服务器的，直接从本地缓存中取。但 F5 会忽视缓存（所以使用fiddler 之类的 http 协议监察工具时，不要 F5 误认为 Expires 和 max-age 是无效的）。


其余存储技巧

1. cookie

兼容没有问题， 十分容易操作，但cookie存储有限，加上cookie每次都要上传到服务器，浪费了带宽。

2. web storage

基于HTML5的webStorage可能能给你带的新的希望，而且最大支持不超过5MB的数据存储。可惜的是，IE方面只支持IE8以上的版本。

						
IE	Firefox	Opera	Chrome	Safari	iPhone	Android
8.0+	3.0+	10.5+	4.0+	4.0+	2.0+	2.0+

sessionStorage用于本地存储一个会话（ session ）中的数据，这些数据只有在同一个会话中的页面才能访问并且当会话结束后数据也随之销毁。因此sessionStorage不是一种持久化的本地存储，仅仅是会话级别的存储。

localStorage用于持久化的本地存储，除非主动删除数据，否则数据是永远不会过期的。

相关实例

如下图，您可以看到左侧一个高高的（如果你是首次进入）的垂直菜单栏，长相如下：



想达到的效果是你刷新页面，该列表项依旧处于收起状态



有意思吧？怎么玩的呢？先看看京东商城的做法。
参考京东商城个人中心的菜单栏设计的；京东的记录用户展开收起细则的做法是使用 cookie，所有浏览器都使用cookie，一个关键字为myjd的cookie，如下截图所示（FireFox6下）：



这是很OK的做法，客户端记录一些可以丢失的数据，且大小不大的时候，cookie是首选，毕竟，所有的浏览器都鸟它。

3、 变量

```
var array = [];  
var length = array.length;  
for(var i=0, i<len; i++)  
  
//固定变量  
var STATIC_VALUE = "coverguo so handsome!";
```