


# HTTP 缓存与缓存更新方案

 zenglinan commented on 15 Dec 2019

## HTTP 缓存

1. 强缓存
2. 协商缓存
3. 用户行为对缓存的影响
4. 如何保证客户端及时更新缓存
5. 总体缓存方案

### 1. 强缓存

强缓存是指直接通过本地缓存获取资源，不用经过服务器

常用字段：

- expires

值为一个绝对时间的 GMT 格式的时间字符串，如果发送请求的时间在 expires 之前，那么本地缓存有效，否则就会发送请求到服务器来获取资源。

缺点：无法保证客户端按照标准时间设定

- Cache-Control(常用值如下):
  - max-age：允许的最大缓存秒数
  - no-store：不允许使用缓存，**每次都要向服务器获取**
  - no-cache：不允许使用本地缓存，**每次都要向服务器进行协商缓存**
  - public：允许被**所有中间代理和终端**浏览器缓存
  - private：只允许被终端浏览器缓存

Cache-Control 比 expires 优先级高

### 2. 协商缓存

协商缓存是指客户端**向服务端确认资源是否用**

常用字段：

- Last-Modified / If-Modified-Since：

值是 GMT 格式的时间字符串，具体流程如下：

1. 浏览器第一次请求资源，服务端返回 Last-Modified，表示资源在服务端的最后修改时间。
2. 浏览器第二次请求的时候会在请求头上携带 If-Modified-Since，值为上次返回的 Last-Modified
3. 服务端收到请求后，比较保存的 Last-Modified 和 请求报文中的 If-Modified-Since，如果一致就返回 304 状态码，不一致就返回新资源，同时更新 Last-Modified 值

- ETag / If-None-Match

值是服务器生成的资源标识符，当资源修改后这个值会被改变，

具体流程与 Last-Modified、If-Modified-Since 相似，但与 Last-Modified 不一样的是，当服务器返回304的响应时，由于 ETag 重新生成过，response header中还会把这个 ETag 返回，即使这个 ETag 跟之前的没有变化。

## 既生 Last-Modified 何生 Etag

- 一些文件也许会周期性的更改，但是他的内容并不改变(仅仅改变的修改时间)，这个时候我们并不希望客户端认为这个文件被修改了
- 某些文件修改非常频繁，比如在秒以下的时间内进行修改，(比方说1s内修改了N次)，If-Modified-Since能检查到的粒度是s级的，这种修改无法判断

## 3. 用户行为对缓存的影响

用户操作	Expires/Cache-Control	Last-Modified/Etag
------	-----------------------	--------------------

## 4. 如何保证客户端及时更新缓存

通过更新页面中引用的资源路径，让浏览器主动放弃缓存，加载新资源。常用更新方法如下：

### 1. 用数字做版本号

```
<link href="a.css?v=1.0.1">
```

但是如果使用版本号的话，假如每次只更新一个文件，其他未更改的文件为了保持版本一致都需要更新，所以不推荐用版本号。

### 2. 用文件摘要做版本号

另一种办法就是采用**数据摘要要算法**对文件求摘要值，拼接到路径参数上，更新某一文件时，只需更新对应文件的路径即可。如下：

```
<link href="a.css?v=0cadf2356">
```

但这样有一个问题：

更新对应文件时，html 中的引用路径也需要修改，**先部署页面？还是先部署静态资源？**

**先部署页面**，在两者部署间隔中，假如有用户访问新页面，此时**页面中的资源路径已经更新，但资源仍未更新**。客户端看到这个新的路径，就会发起请求，然而依然会拿到旧资源并作为缓存，并且，即便后面资源更新了，在已拿到资源缓存过期前，一直都不会重新请求，**一直都是旧资源**

**先部署资源**，在两者部署间隔中，假如有用户访问，此时 html 中资源路径仍未更新，资源已经更新。这时假如本地缓存未过期，则不会请求，一切安好。假如这时本地缓存过期了，会去协商缓存，就会出现**旧 html 引用新资源的情况，可能会导致页面出错**。

比如：

```
// 旧 html
<div class="aaa">...</div>
// 旧 css 资源
.aaa{ ... }

// 新 html
<div class="bbb">...</div>
// 新 css 资源
.bbb{ ... }
```

这时候旧的 html 引用新资源就会导致页面样式丢失。

所以，这种**覆盖式发布**的方式也不推荐，最好的办法是采取下面的**非覆盖式发布**。

3. 将文件摘要作为资源路径的一部分

**将文件的摘要值作为路径的一部分或者文件名的一部分，而不是参数。**如下：

```
<link href="0cadf2356/a.css">
<link href="0cadf2356_a.css">
```

然后，每次只需要**先部署资源，再部署页面**即可。

## 5. 总结一下缓存及更新方案

1. 本地缓存设置很长的过期时间
2. 采用内容摘要作为缓存更新依据
3. 静态资源CDN部署
4. 需要更新时，更资源发布路径实现非覆盖式发布

参考：[大公司里怎样开发和部署前端代码？——张云龙的回答](#)

