NGINX
Part of F5

nginx-cn.net

NGINX开源社区基础培训系列课程（第三季）

第一讲：NGINX如何控制客户端缓存

主讲人：陶辉

关注“NGINX开源社区”公众号参与后继活动。

NGINX开源社区群

NGINX开源社区公众号

NGINX

第三季：深入剖析HTTP缓存

第三季目录: 深入剖析HTTP缓存

- Nginx如何控制客户端缓存?
- Nginx缓存使用中易陷入的误区
- Nginx缓存的淘汰算法
- Nginx缓存的工作原理

Nginx如何控制客户端缓存?

- 问题
  1. 如果不使用Etag指纹，浏览器还会开启缓存吗?
  2. 如果Nginx返回404，可以让浏览器缓存这个结果吗?
  3. 浏览器会缓存共享资源吗?
  4. 时间与指纹分开使用时，浏览器缓存可以正常工作吗?
  5. Nginx中可以根据哪些因素设置浏览器缓存的过期时间?
  6. Cache-Control中的no-cache、no-store到底有何差别?
  7. 如何让Nginx禁止浏览器缓存资源?
  8. 多用户并发修改资源时，如何防止误覆盖?
  9. If-Range在Range请求中有什么作用?
  10. 对于更新缓慢又很重要的资源，怎样避免使用过期缓存?

RFC7232：指纹

- **响应头部**
  - ETag = entity-tag
    - 仅对同一个URL下的比较有意义
- **请求头部**
  - If-Match = "*" / 1#entity-tag
    - 用于并发修改资源（POST/PUT/DELETE）时，防止误覆盖
  - If-None-Match = "*" / 1#entity-tag
    - 用于过期缓存的更新

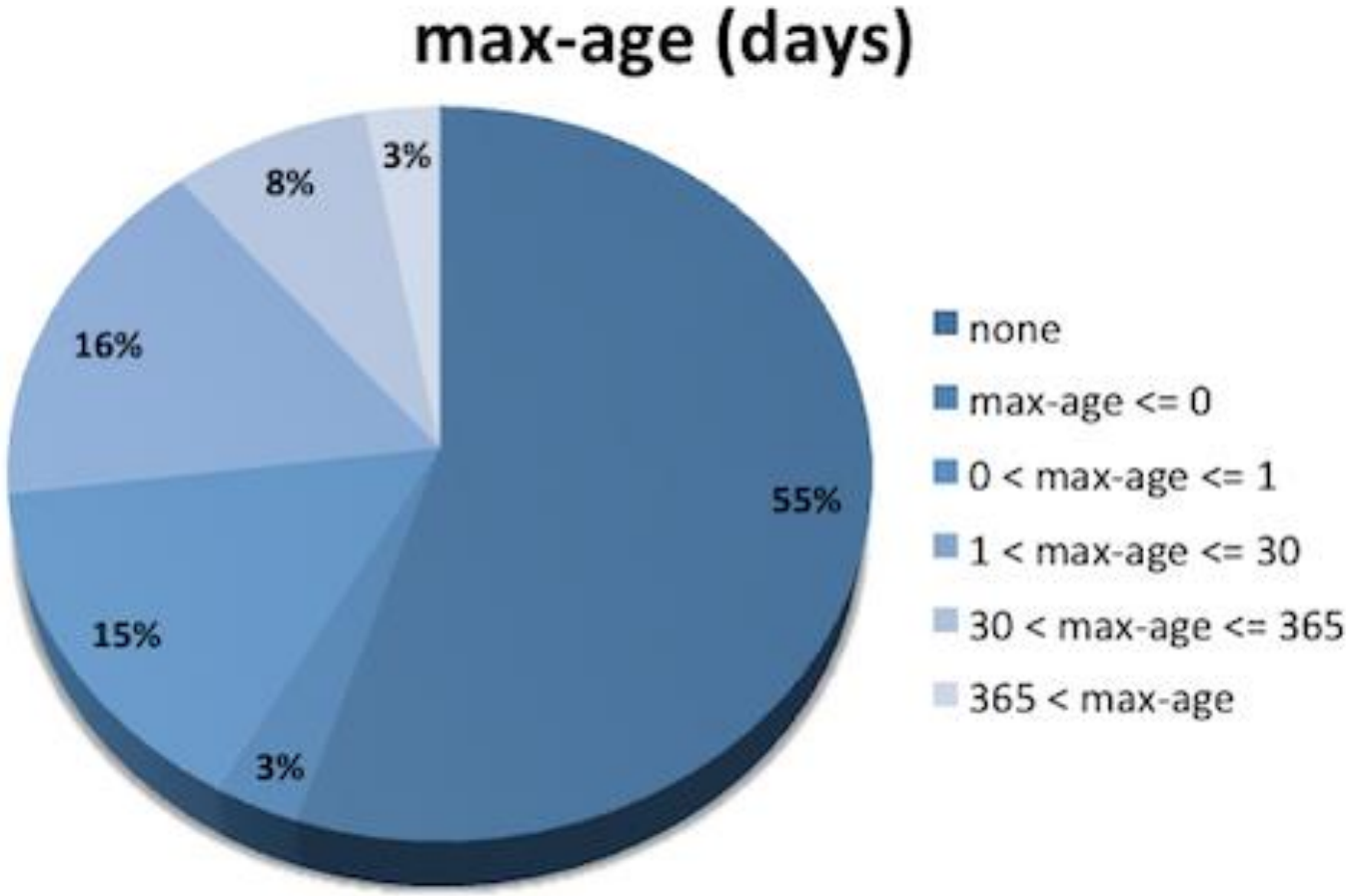
如果不生成Etag指纹，浏览器会不会缓存?

Syntax:	**etag** on | off;
Default:	etag on;
Context:	http, server, location

生成规则：

```
ngx_sprintf(etag->value.data, "\"%xT-%xO\"",
                              r->headers_out.last_modified_time,
                              r->headers_out.content_length_n)
```

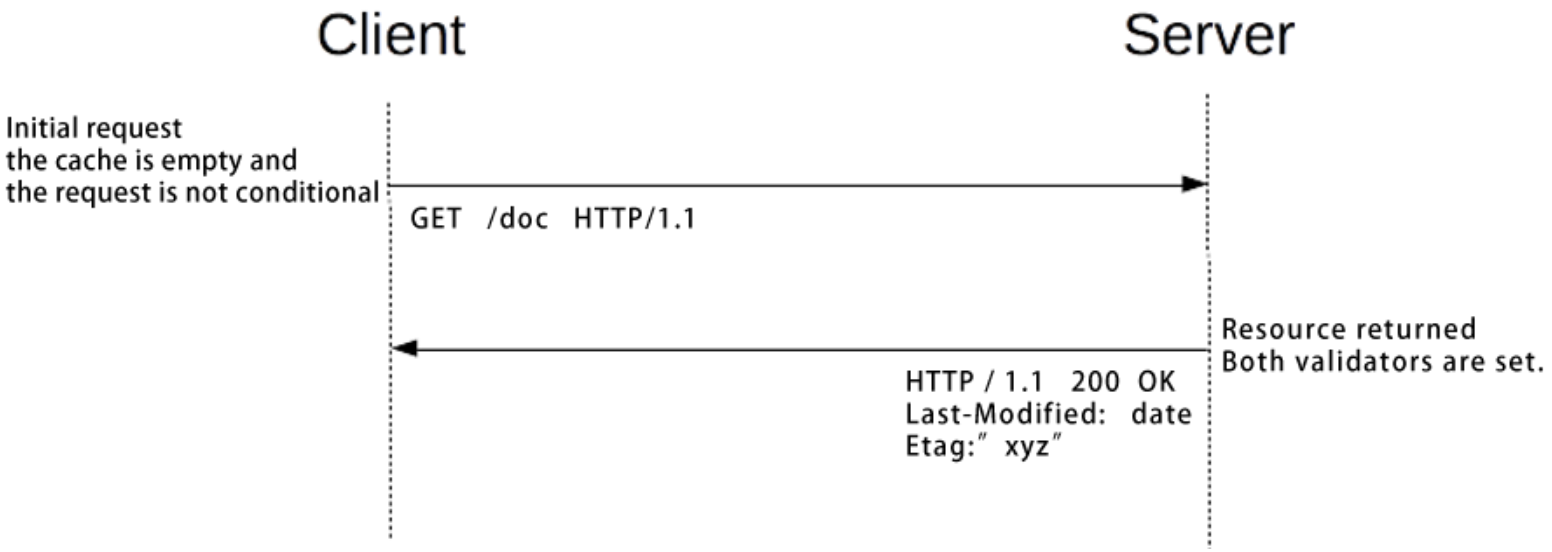
为什么要预估缓存时间?


max-age (days)
55%
3%
15%
16%
8%
3%
none
max-age <= 0
0 < max-age <= 1
1 < max-age <= 30
30 < max-age <= 365
365 < max-age

常见的预估时间

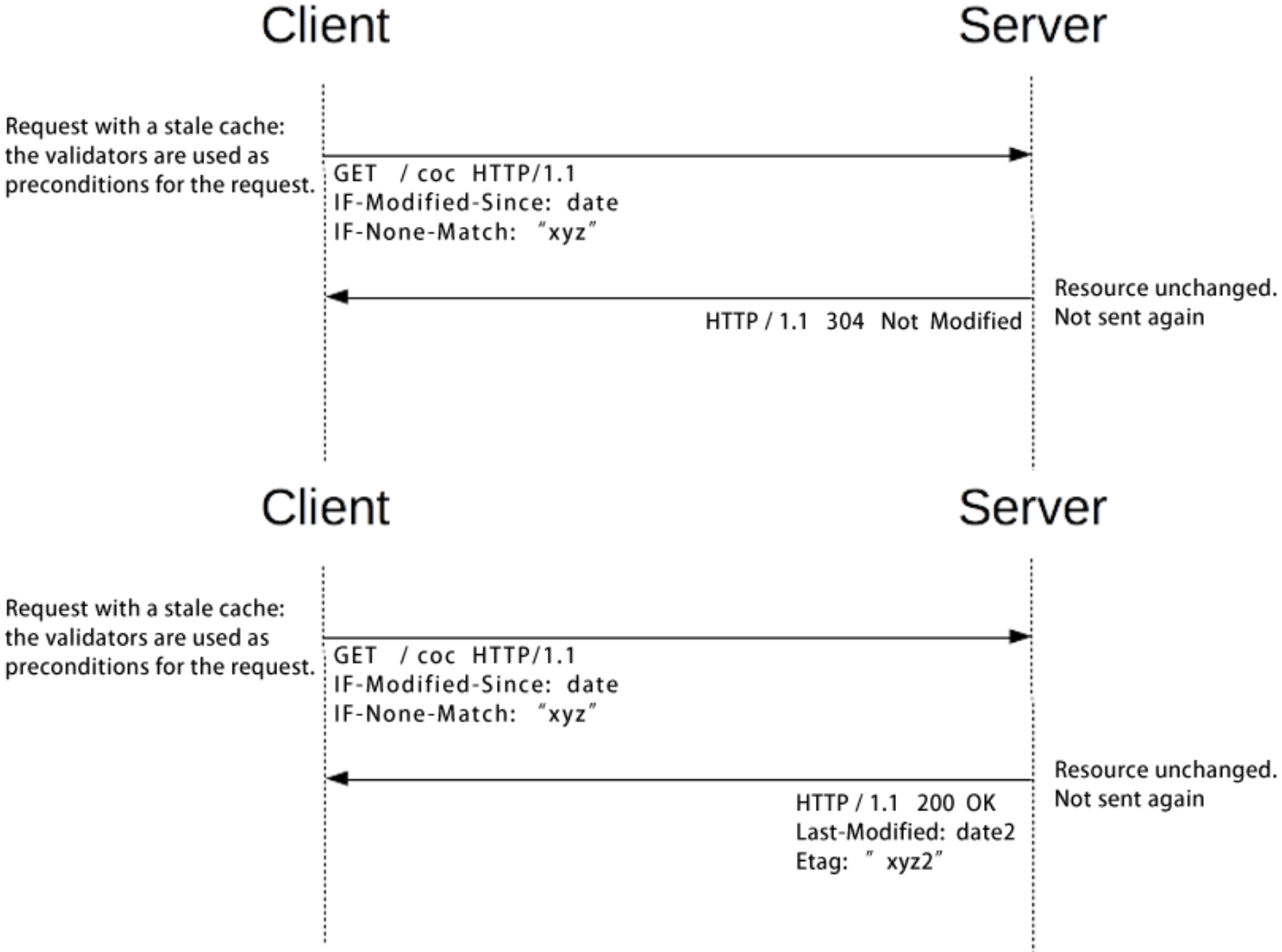
- RFC7234推荐：（DownloadTime– LastModified)*10%

首次缓存

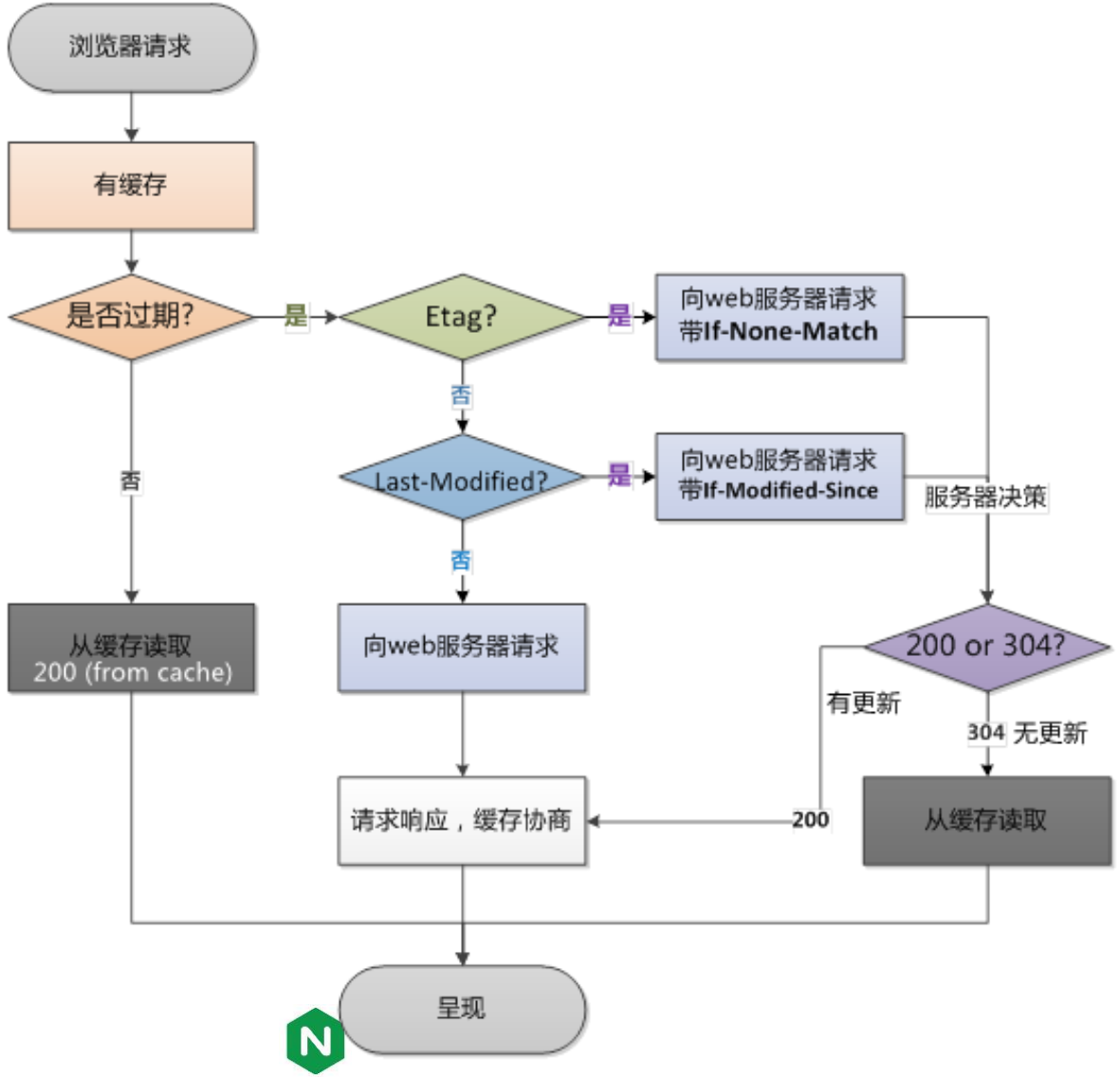

Client
Server
Initial request
the cache is empty and
the request is not conditional
GET /doc HTTP/1.1
Resource returned
Both validators are set.
HTTP / 1.1 200 OK
Last-Modified: date
Etag:" xyz"

基于过期缓存提高网络效率

- 304 Not Modified



浏览器缓存示意图


浏览器请求
有缓存
是否过期?
是
Etag?
是
向web服务器请求
带If-None-Match
否
Last-Modified?
是
向web服务器请求
带If-Modified-Since
服务器决策
否
否
从缓存读取
200 (from cache)
向web服务器请求
200 or 304?
有更新
304 无更新
请求响应，缓存协商
200
从缓存读取
呈现

404响应可以缓存吗？

- RFC7234
  - 请求方法可以被缓存理解（不只于 GET 方法）
  - 响应码可以被缓存理解（404、206 也可以被缓存）
  - 响应与请求的头部没有指明 no-store
  - 响应中至少应含有以下头部中的 1 个或者多个：
    - Expires、max-age、s-maxage、public
    - 当响应中没有明确指示过期时间的头部时，如果响应码非常明确，也可以缓存
  - 如果缓存在代理服务器上
    - 不含有 private
    - 不含有 Authorization

ngx_http_headers_filter_module模块

Syntax:	add_header name value [always];
Default:	-;
Context:	http, server, location, if in location

浏览器会缓存共享资源吗?

- **私有缓存**
  - 仅供一个用户使用的缓存
    - 浏览器
- **共享缓存**
  - 可以供多个用户的缓存，存在于网络中负责转发消息的代理服务器（对热点资源常使用共享缓存，以减轻源服务器的压力，并提升网络效率）
    - 正向代理
    - 反向代理

指纹与时间可以单独生效吗？

- **响应头部**
  - Last-Modified = HTTP-date
- **请求头部**
  - If-Modified-Since = HTTP-date
    - 没有指令时，可用于更新过期缓存
  - If-Unmodified-Since = HTTP-date
    - 用于并发修改资源（POST/PUT/DELETE）时，防止误覆盖

not_modified过滤模块

Syntax:	**if_modified_since** off | exact | before;
Default:	if_modified_since exact;
Context:	http, server, location

- **off**
  - 忽略请求中的if_modified_since头部，即除非比较指纹（且请求中没有携带if-modified-since头部），否则不会返回304
- **exact**
  - 精确匹配if_modified_since头部与last_modified的值
- **before**
  - 若if_modified_since大于等于last_modified的值，则返回304

缓存多久？ -- RFC7234

- 响应头部
  - Expires = http-date
  - Pragma = 1#pragma-directive
    - pragma-directive = "no-cache" / extension-pragma
      - extension-pragma = token ["=" (token / quoted-string)]
- 单向Cache-Control

Cache-Control 头部

- **Cache-Control = 1#cache-directive**
  - cache-directive = token ["=" (token / quoted-string)]
    - delta-seconds = 1*DIGIT
      - RFC 规范中的要求是，至少能支持到 2147483648 (2^31)
- **请求中的头部：** max-age、max-stale、min-fresh、no-cache、no-store、no-transform、only-if-cached
- **响应中的头部：** max-age、s-maxage 、 must-revalidate 、proxy-revalidate 、no-cache、no-store、no-transform、public、private

Cache-Control 头部在请求中的值

- **max-age**：不想要在**代理服务器**中缓存了太长时间（>max-age seconds）的资源
- **max-stale**：可以接受**代理服务器**上的过期缓存。若 max-stale 后没有值，则表示无论过期多久客户端都可使用
- **min-fresh**：要求**服务器**使用其缓存时，保证至少在min-fresh秒内不会过期
- **no-cache**：告诉代理服务器，不能直接使用已有缓存作为响应返回，除非带着缓存条件到上游服务端得到 304 验证返回码才可使用现有缓存
- **no-store**：告诉各代理服务器不得缓存这个请求及其响应
- **no-transform**：告诉代理服务器不要修改消息包体的内容
- **only-if-cached**：告诉代理服务器仅能返回缓存，没有缓存的话就返回 504

Cache-Control 头部在响应中的值

- **must-revalidate**：告诉客户端一旦缓存过期，必须向服务器验证后才可使用
- **proxy-revalidate**：与 must-revalidate 类似，但它仅对代理服务器的共享缓存有效
- **no-cache**：告诉客户端不能直接使用缓存的响应，使用前必须在源服务器验证得到 304 返回码。如果 no-cache 后指定头部，则若客户端的后续请求及响应中不含有这些头则可直接使用缓存
- **max-age**：告诉客户端缓存 Age 超出 max-age 秒后则缓存过期

Cache-Control 头部在响应中的值

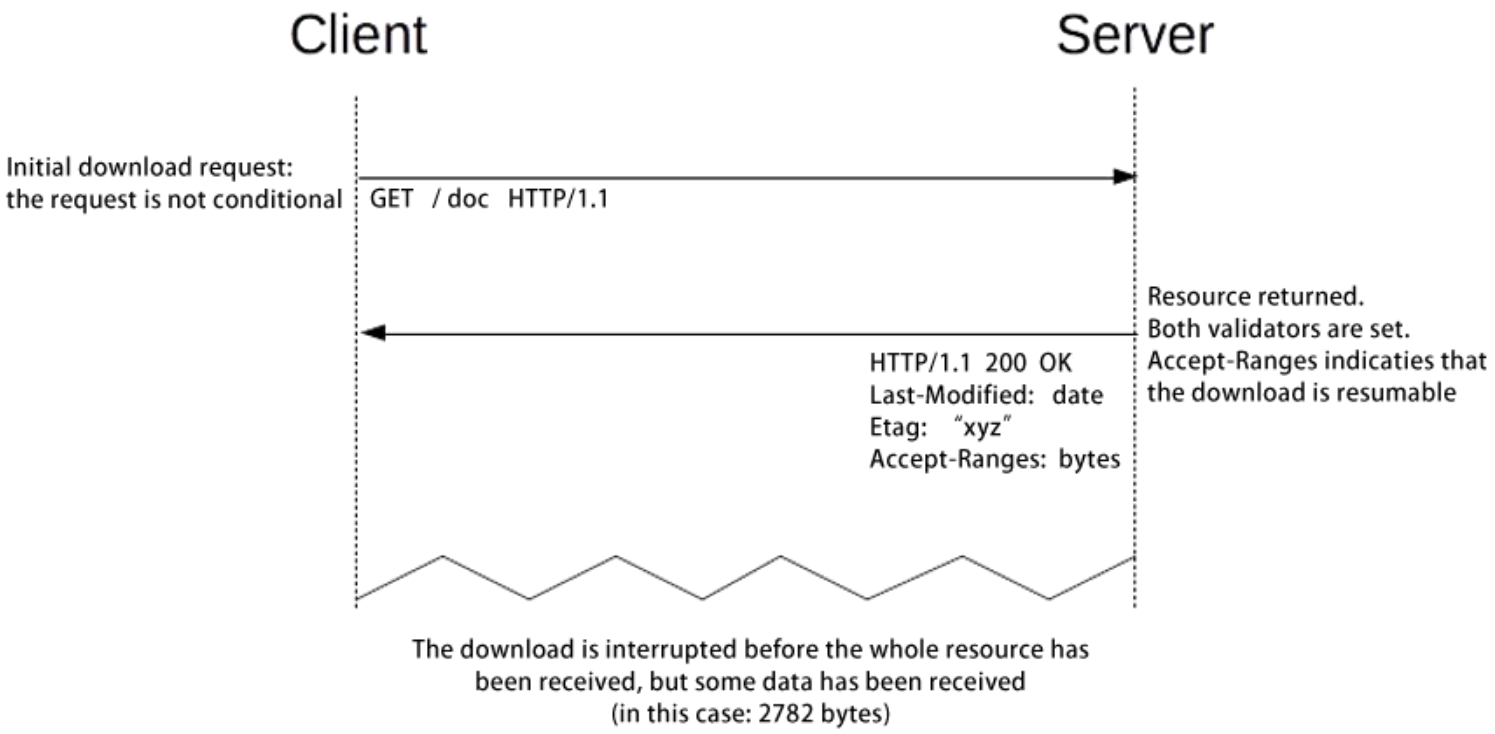
- **s-maxage**：与 max-age 相似，但仅针对共享缓存，且优先级高于 max-age 和 Expires
- **public**：表示无论私有缓存或者共享缓存，皆可将该响应缓存
- **private**：表示该响应不能被代理服务器作为共享缓存使用。若 private 后指定头部，则在告诉代理服务器不能缓存指定的头部，但可缓存其他部分
- **no-store**：告诉所有下游节点不能对响应进行缓存
- **no-transform**：告诉代理服务器不能修改消息包体的内容

expires指令

Syntax:	**expires** [modified] *time*; **expires** epoch | max | off;
Default:	expires off;
Context:	http, server, location, if in location

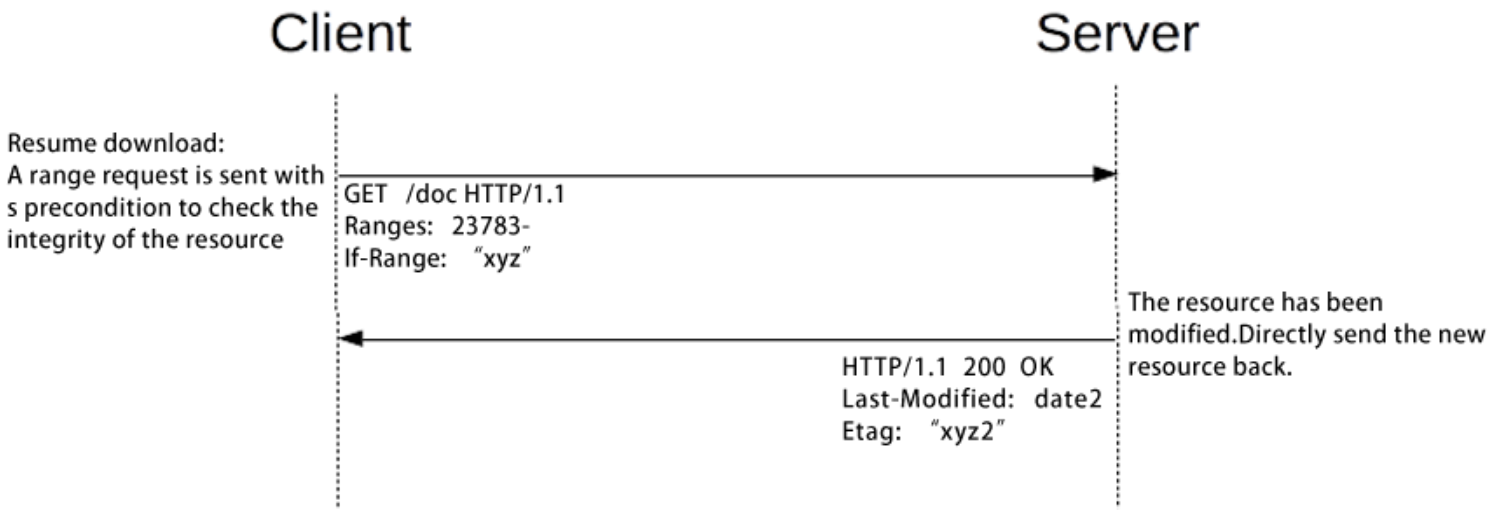
- **max:**
  - Expires: Thu, 31 Dec 2037 23:55:55 GMT
  - Cache-Control: max-age=315360000（10年）
- **off：不添加或者修改Expires和Cache-Control字段**
- **epoch：**
  - Expires: Thu, 01 Jan 1970 00:00:01 GMT
  - Cache-Control: no-cache
- **time：设定具体时间，可以携带单位**
  - 一天内的具体时刻可以加@，比如下午六点半：@18h30m
    - 设定好Expires，自动计算Cache-Control
    - 如果当前时间未超过当天的time时间，则Expires到当天time，否则是第二天的time时刻
  - 正数
    - 设定Cache-Control时间，计算出Expires
  - 负数
    - Cache-Control: no-cache ，计算出Expires

Range请求中的缓存（1）



Range请求中的缓存（2）

- If-Range = entity-tag / HTTP-date



谢谢

NGINX
Part of F5

近期活动推荐

New

NGINX 公开课：NGINX Plus 回归本源系列

09月23日　使用 NGINX / NGINX Plus 构建CDN
10月21日　使用 NGINX / NGINX Plus 构建K8S Ingress Controller
11月25日　使用 ModSec / App Protect 模块构建NGINX WA

每周三 | 14:00-15:00
讲师：邹俊 NGINX大中华区解决方案架构师

扫码报名

NGINX开源社区基础培训系列课程（第三季）

- 深入剖析NGINX HTTP缓存

课程安排：每周四，晚8:00-9:00

- 8月06日 NGINX如何控制客户端缓存？
- 8月13日 NGINX缓存使用中易陷入的误区
- 8月20日 NGINX缓存的淘汰算法
- 8月27日 NGINX缓存的工作原理

讲师：陶辉
NGINX顶级专家

NGINX
Part of F5

扫码进入直播间或打开ZOOM APP
网络研讨会ID：939 6325 1302
课程口令密码：666

扫码关注NGINX社区

扫码加入NGINX社区官方微信群

NGINX
Part of F5

关注我们

NGINX开源社区微信

NGINX 社区微信群

NGINX开源社区官方微博

NGINX开源社区是F5/NGINX面向所有NGINX用户的官方社区。我们秉持“开放，包容，沟通，贡献（Open，Inclusive，Connect，Contribution）”之宗旨，与业界共建开放、包容、活跃的“NGINX用户之家”；秉承开源的精神，在社区治理上高度开放，为所有NGINX的用户，开发者和技术爱好者，提供一个方便学习、讨论的场所。也期待您成为此社区中活跃的一员，贡献您的文章，博客，代码，踊跃讨论与回答问题，打造您个人品牌和影响力。

点击访问NGINX开源社区网站：nginx-cn.net