

暂停

01:23 / 10:00

http 状态码

- ◆ 状态码分类
- ◆ 常见状态码
- ◆ 关于协议和规范

imooc

 慕课网

状态码分类 - 1

- ◆ 1xx 服务器收到请求
- ◆ 2xx 请求成功，如 200
- ◆ 3xx 重定向，如 302

imooc

 慕课网

03:16 / 10:00

状态码分类 - 2

- ◆ 4xx 客户端错误，如 404
- ◆ 5xx 服务端错误，如 500

 慕课网

常见状态码 - 1

- ◆ 200 成功
- ◆ 301 永久重定向（配合 location，浏览器自动处理）
- ◆ 302 临时重定向（配合 location，浏览器自动处理）

 慕课网

常见状态码 - 2

- ◆ 304 资源未被修改
- ◆ 404 资源未找到
- ◆ 403 没有权限

常见状态码 - 3

- ◆ 500 服务器错误
- ◆ 504 网关超时

http methods

- ◆ 传统的 methods
- ◆ 现在的 methods
- ◆ Restful API



传统的 methods

- ◆ get 获取服务器的数据
- ◆ post 像服务器提交数据
- ◆ 简单的网页功能，就这两个操作



现在的 methods - 1

- ◆ get 获取数据
- ◆ post 新建数据
- ◆ patch/put 更新数据

 慕课网

现在的 methods - 2

- ◆ delete 删除数据

 慕课网

Restful API

- ◆ 一种新的 API 设计方法（早已推广使用）
- ◆ 传统 API 设计：把每个 url 当做一个功能
- ◆ Restful API 设计：把每个 url 当做一个唯一的资源

 慕课网

05:32 / 11:46

如何设计成一个资源？

- ◆ 尽量不用 url 参数
- ◆ 用 method 表示操作类型

 慕课网

暂停

07:32 / 11:46

不使用 url 参数

- ◆ 传统 API 设计 : `/api/list?pageIndex=2` 当作函数, 参数是 `pageIndex=2`
- ◆ Restful API 设计 : `/api/list/2` 当作一个资源标识, 具体操作通过 methods

 慕课网

用 method 表示操作类型 (传统 API 设计)

- ◆ post 请求 `/api/create-blog`
- ◆ post 请求 `/api/update-blog?id=100`
- ◆ get 请求 `/api/get-blog?id=100`

 慕课网

用 method 表示操作类型 (Restful API 设计)

- ◆ post 请求 /api/blog
- ◆ patch 请求 /api/blog/100
- ◆ get 请求 /api/blog/100

 慕课网

http headers

- ◆ 常见的 Request Headers
- ◆ 常见的 Response Headers

 慕课网

Request Headers

- ◆ Accept 浏览器可接收的数据格式
- ◆ Accept-Encoding 浏览器可接收的压缩算法，如 gzip
- ◆ Accept-Language 浏览器可接收的语言，如 zh-CN

Request Headers

- ◆ Connection: keep-alive 一次 TCP 连接重复使用
- ◆ cookie 请求时浏览器自动带上
- ◆ Host 请求时的域名

Request Headers

- ◆ User-Agent (简称 UA) 浏览器信息
- ◆ Content-type 发送数据的格式, 如 application/json

Response Headers

- ◆ Content-type 返回数据的格式, 如 application/json
- ◆ Content-length 返回数据的大小, 多少字节
- ◆ Content-Encoding 返回数据的压缩算法, 如 gzip

Response Headers

- ◆ Set-Cookie

缓存相关的 headers

- ◆ Cache-Control Expires
- ◆ Last-Modified If-Modified-Since
- ◆ Etag If-None-Match

http 缓存

- ◆ 关于缓存的介绍
- ◆ http 缓存策略（强制缓存 + 协商缓存）
- ◆ 刷新操作方式，对缓存的影响

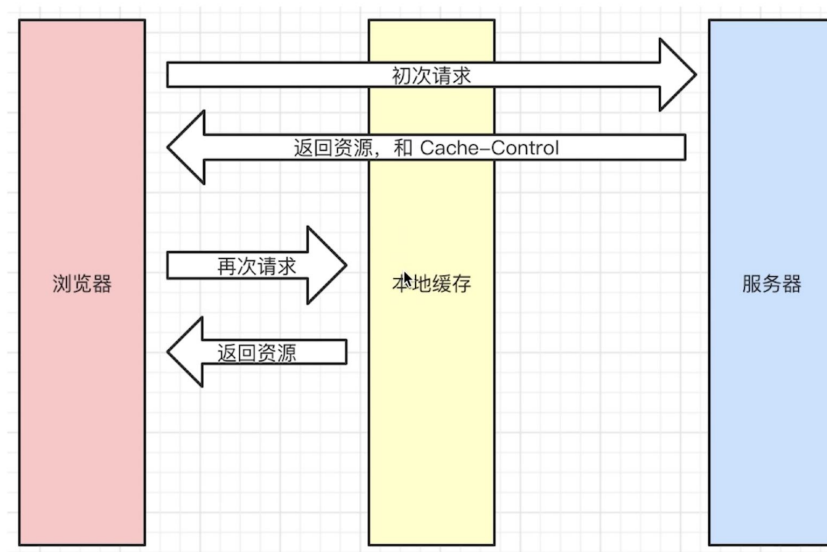
关于缓存

- ◆ 什么是缓存？
- ◆ 为什么需要缓存？
- ◆ 哪些资源可以被缓存？— 静态资源（js css img）

暂停

03:45 / 10:06

http 缓存 - 强制缓存




慕课网

Cache-Control

- ◆ Response Headers 中
- ◆ 控制强制缓存的逻辑
- ◆ 例如 Cache-Control: max-age=31536000 (单位是秒)

慕课网

http 缓存 - 强制缓存

Name	Status	T...	I...	Size	Time
 1701fb5290f89384?ima...	200	d...	O...	(disk cache)	3 ms

慕课网

暂停

08:04 / 10:06

cache-control 的值

- ◆ max-age 资源的最大过期时间
- ◆ no-cache 不用本地缓存，正常去服务端请求
- ◆ no-store 不用本地缓存，并且不用服务端的缓存措施

慕课网

cache-control 的值

- ◆ max-age
- ◆ private 只允许最终用户作为缓存
- ◆ no-cache
- ◆ public 允许中间路由或者代理作为缓存
- ◆ no-store

关于 Expires

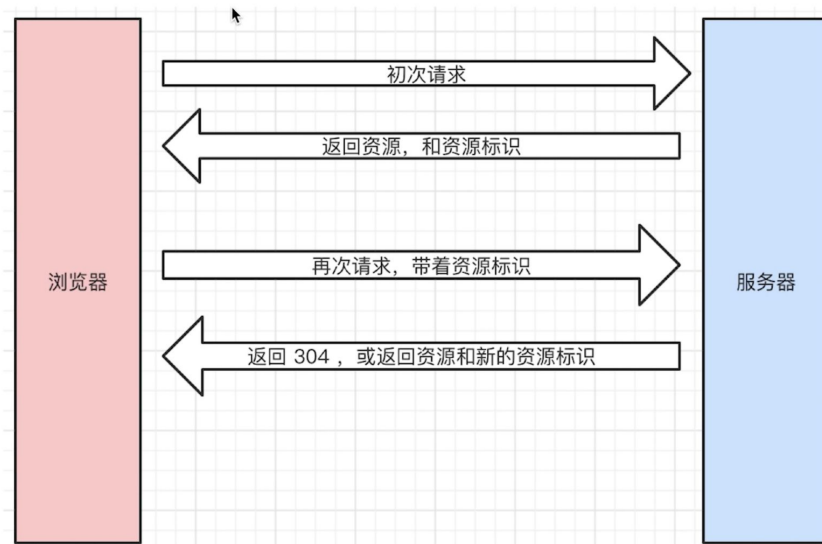
- ◆ 同在 Response Headers 中
- ◆ 同为控制缓存过期
- ◆ 已被 Cache-Control 代替

http 缓存 - 协商缓存

- ◆ 服务器端缓存策略
- ◆ 服务器判断客户端资源，是否和服务端资源一样
- ◆ 一致则返回 304，否则返回 200 和最新的资源

慕课网

http 缓存 - 协商缓存



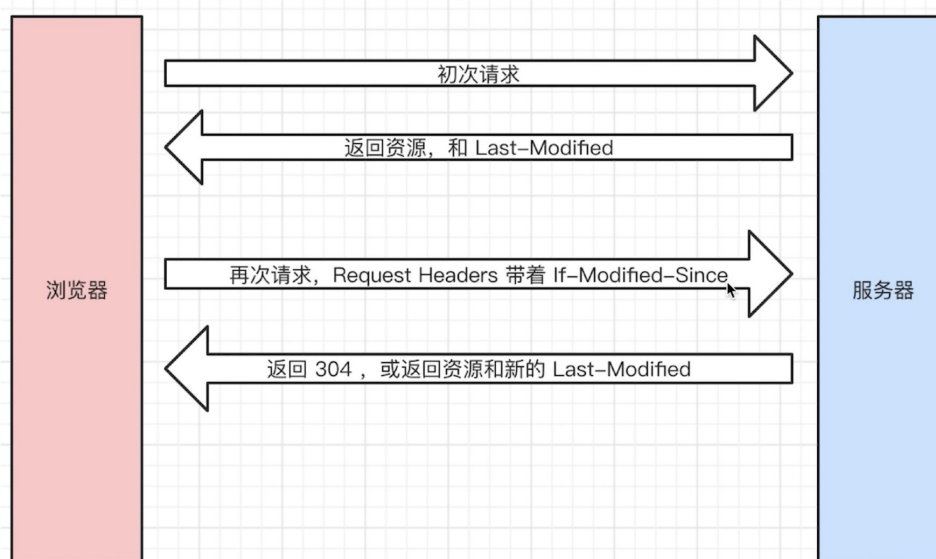
慕课网

资源标识

- ◆ 在 Response Headers 中，有两种
- ◆ Last-Modified 资源的最后修改时间
- ◆ Etag 资源的唯一标识（一个字符串，类似人类的指纹）

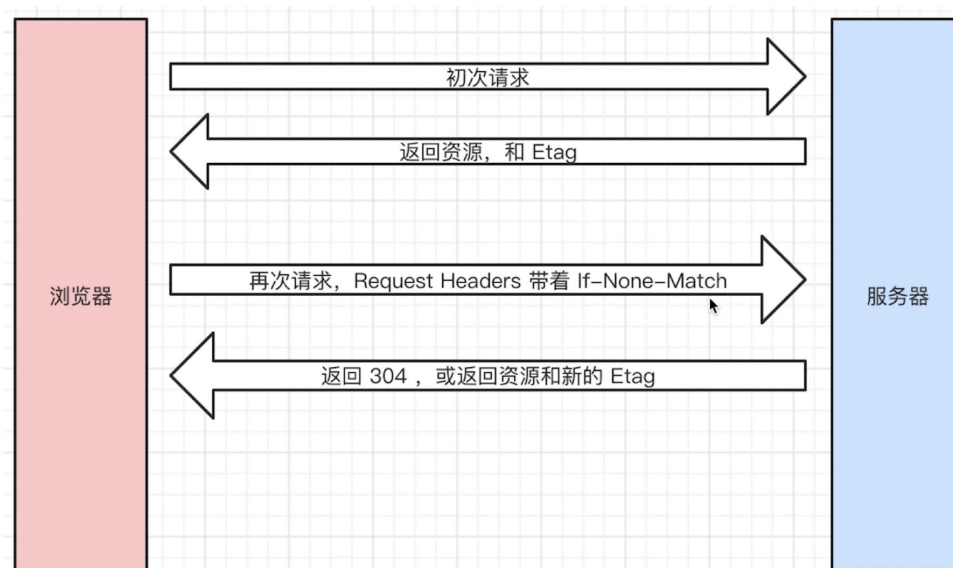
慕课网

Last-Modified



慕课网

Etag



慕课网

Headers 示例

Response Headers

Date: Sun, 02 Feb 2020 05:33:03 GMT**Etag:** 7d5e319a15811687457696629e**Etag:** "6bd9NvC2BFM:52726"**HitInfo:** CDN_HIT**Last-Modified:** Mon, 30 Dec 2019 17:42:38 GMT**Server:** DnionOS

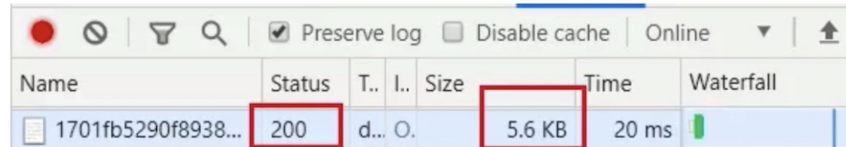
Request Headers

Connection: keep-alive**Cookie:** _ga=GA1.2.1844438941.1568202998; _ntes_10144b72eba172718,1572789863971**Host:** www.dnion.com**If-Modified-Since:** Mon, 30 Dec 2019 17:42:38 GMT**If-None-Match:** "6bd9NvC2BFM:52726"

慕课网

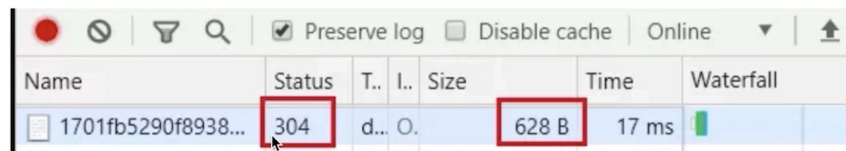
请求示例

第一次访问:



Name	Status	T..	I..	Size	Time	Waterfall
1701fb5290f8938...	200	d...	O.	5.6 KB	20 ms	

第二次访问:



Name	Status	T..	I..	Size	Time	Waterfall
1701fb5290f8938...	304	d...	O.	628 B	17 ms	

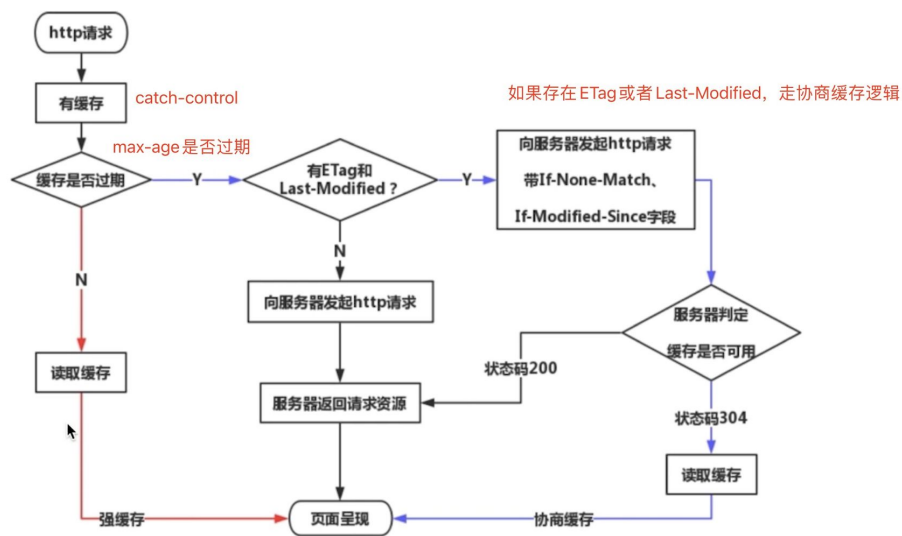
慕课网

Last-Modified 和 Etag

- ◆ 会优先使用 Etag
- ◆ Last-Modified 只能精确到秒级
- ◆ 如果资源被重复生成, 而内容不变, 则 Etag 更精确

慕课网

http 缓存 - 综述



慕课网

三种刷新操作

- ◆ 正常操作：地址栏输入 url，跳转链接，前进后退等
- ◆ 手动刷新：F5，点击刷新按钮，右击菜单刷新
- ◆ 强制刷新：ctrl + F5

慕课网

不同刷新操作，不同的缓存策略

- ◆ 正常操作：强制缓存有效，协商缓存有效
- ◆ 手动刷新：强制缓存失效，协商缓存有效
- ◆ 强制刷新：强制缓存失效，协商缓存失效