浏览器缓存和数据压缩

HTTP 缓存机制分类

1. 200 from cache：直接从本地缓存中获取响应，最快速，最省流量，因为根本没有向服务器发送请求
2. 304 Not Modified：协商缓存，浏览器在本地没有命中的情况下，请求头中发送一定的校验数据到服务端，如果服务端的数据没有改变，浏览器从本地缓存响应，返回 304。特点：快速，发送的数据很少，只返回一些基本的响应头信息，数据量很小，不发送实际响应体
3. 200 OK：以上两种缓存全部失败，服务器返回完整响应。没有用到缓存，相对最慢

header 设置 HTTP 缓存机制

1. pragma：HTTP1.0 时代的遗留产物，该字段被设置为 no-cache 时，会告知浏览器禁用本地缓存，即每次都向服务器发送请求
2. Expires：HTTP1.0 时代用来启用本地缓存的字段，设置值如 ‘Thu, 31 Dec 2037 23:55:55 GMT’ 的格林威治的时间。但浏览器与服务器的时间无法保持一致，如果差距大就会影响缓存结果
3. Cache-Control：HTTP1.1 针对 Expires 时间不一致的解决方案，运用 Cache-Control 告知浏览器缓存过期的时间间隔而不是时刻，即使具体时间不一致，也不影响缓存的管理

优先级：Pragma > Cache-Control > Expires

Cache-Control 配置

1. no-store：禁止浏览器缓存响应
2. no-cache：不允许直接使用本地缓存，先发起请求和服务器协商
3. max-age=delta-seconds：告知浏览器该响应本缓存的有效的最长期限，以秒为单位

协商缓存

1. 当浏览器没有命中本地缓存，如本地缓存过期或者响应中声名不允许直接使用本地缓存，那么浏览器肯定会发起服务端请求
2. 服务端会验证数据是否修改，如果没有就通知浏览器使用本地缓存

header 设置协商缓存

1. Last-Modified：通知浏览器资源的最后修改时间，设置值如 ‘Thu, 31 Dec 2037 23:55:55 GMT’ 的格林威治的时间
2. If-Modified-Since：得到资源的最后修改时间后，会将这个信息通过 If-Modified-Since 提交到服务器做检查，如果没有修改，返回 304 状态码，设置值如 ‘Thu, 31 Dec 2037 23:55:55 GMT’ 的格林威治的时间
3. ETag：HTTP1.1 推出，文件的指纹标识符，如果文件内容修改，指纹也会改变，设置值如 ‘5a643fc7-38a3’
4. If-None-Match：本地缓存失效，会携带此值去请求服务端，服务端判断该资源是否改变，如果没有改变，直接使用本地缓存，返回 304