缓存

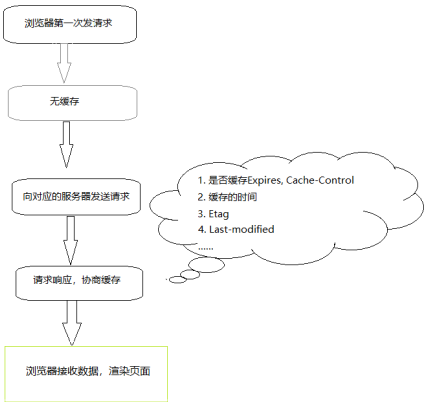
1. 缓存理解

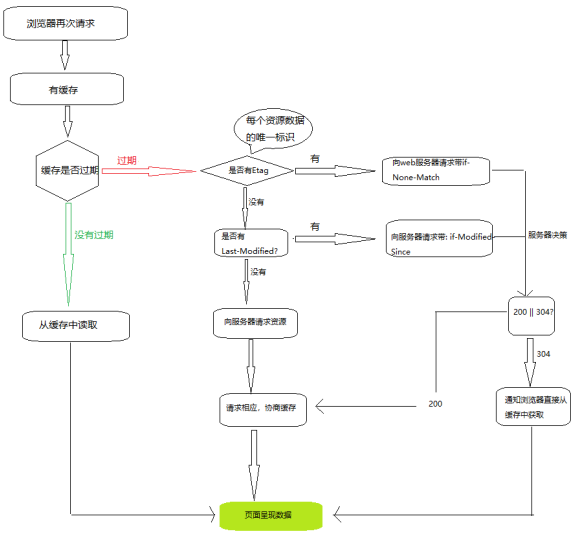
1. 缓存定义:
    1. 浏览器在本地磁盘上将用户之前请求的数据存储起来，当访问者再次需要改数据的时候无需再次发送请求，直接从浏览器本地获取数据
2. 缓存的好处:
    1. 减少请求的个数
    2. 节省带宽，避免浪费不必要的网络资源
    3. 减轻服务器压力
    4. 提高浏览器网页的加载速度，提高用户体验

2. 缓存分类

1. 强缓存
    1. 不会向服务器发送请求，直接从本地缓存中获取数据
    2. 请求资源的的状态码为: 200 ok(from memory cache)
2. 协商缓存
    1. 向服务器发送请求，服务器会根据请求头的资源判断是否命中协商缓存
    2. 如果命中，则返回304状态码通知浏览器从缓存中读取资源
3. 强缓存 & 协商缓存的共同点
    1. 都是从浏览器端读取资源
4. 强缓存 VS 协商缓存的不同点
    1. 强缓存不发请求给服务器
    2. 协商缓存发请求给服务器，根据服务器返回的信息决定是否使用缓存

3. 缓存使用示意图





4. 缓存中的header参数

1、强缓存的header参数（responsHeader）

1. **expires：**
   - 这是http1.0时的规范；它的值为一个绝对时间的GMT格式的时间字符串，如`Mon, 10 Jun 2015 21:31:12 GMT`，如果发送请求的时间在expires之前，那么本地缓存始终有效，否则就会发送请求到服务器来获取资源
2. **cache-control：max-age=number**
   - 这是http1.1时出现的header信息，主要是利用该字段的max-age值来进行判断，它是一个相对值；资源第一次的请求时间和Cache-Control设定的有效期，计算出一个资源过期时间，再拿这个过期时间跟当前的请求时间比较，如果请求时间在过期时间之前，就能命中缓存，否则就不行；
   - cache-control常用的值：
     1. no-cache: 不使用本地缓存，需要使用协商缓存。先与服务器确认返回的响应是否被更改，如果之前的响应中存在Etag，那么请求的时候会与服务器端进行验证，如果资源未被更改则使用缓存。
     2. no-store: 直接禁止浏览器缓存数据，每次用户请求该资源，都会向服务器发送一个请求，每次都会下载完整的资源。
     3. public：可以被所有的用户缓存，包括终端用户和CDN等中间代理服务器。
     4. private：只能被终端用户的浏览器缓存，不允许CDN等中继缓存服务器对其缓存。
3. 注意：当cache-control与Expires共存的时候cache-control的优先级高

2、协商缓存的header参数

重点：协商缓存都是由服务器来确定缓存资源是否可用的，所以客户端与服务器端要通过某种标识来进行通信，从而让服务器判断请求资源是否可以缓存访问

- **Last-Modified / If-Modified-Since**:二者的值都是GMT格式的时间字符串

1. 浏览器第一次跟服务器请求一个资源，服务器在返回这个资源的同时，在respone的header加上Last-Modified，这个header表示这个资源在服务器上的最后修改时间
2. 浏览器再次跟服务器请求这个资源时，在request的header上加上If-Modified-Since，这个header的值就是上一次请求时返回的Last-Modified的值
3. 服务器再次收到资源请求时，根据浏览器传过来If-Modified-Since和资源在服务器上的最后修改时间判断资源是否有变化，如果没有变化则返回304 Not Modified，但是不会返回资源内容；如果有变化，就正常返回资源内容。当服务器返回304 Not Modified的响应时，response header中不会再添加Last-Modified，因为既然资源没有变化，那么Last-Modified也就不会改变，这是服务器返回304时的response header
4. 浏览器收到304的响应后，就会从缓存中加载资源
5. 如果协商缓存没有命中，浏览器直接从服务器加载资源时，Last-Modified的Header在重新加载的时候会被更新，下次请求时，If-Modified-Since会启用上次返回的Last-Modified值

- **Etag / If-None-Match**
  1. 这两个值是由服务器生成的每个资源的唯一标识字符串，只要资源有变化就这个值就会改变
  2. 其判断过程与Last-Modified / If-Modified-Since类似

- **既生Last-Modified何生Etag**
  1. HTTP1.1中Etag的出现主要是为了解决几个Last-Modified比较难解决的问题

2. 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，而重新GET
3. 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(或者说UNIX记录MTIME只能精确到秒)；
4. 某些服务器不能精确的得到文件的最后修改时间。

- 小结：
  - 利用Etag能够更加准确的控制缓存，因为Etag是服务器自动生成或者由开发者生成的对应资源在服务器端的唯一标识符。
  - Last-Modified与ETag是可以一起使用的，服务器会优先验证ETag，一致的情况下，才会继续比对Last-Modified，最后才决定是否返回304。