Effective HTTP Caching — Part I

msingh Follow
Nov 26, 2018 · 4 min read

This series presents an exhaustive dive in the HTTP caching concepts. It is also accompanied by code hosted here which readers can download and run on their own.

Defintion

RFC7234 defines HTTP cache as a local store of response messages and the subsystem that controls storage, retrieval, and deletion of messages in it.

Browsers

Browsers implement HTTP caching, conforming to the HTTP spec.
Use of proper HTTP header directives by your server, to instruct the browser on when and for how long the browser can cache the response, will go a long way in boosting the performance of your application by reducing unnecessary network roundtrips.

Although the HTTP spec allows to cache responses other than successful retrieval (200 OK) of a GET request, in practice most caches typically limit themselves to GET request response.

. . .

Cacheable Response : max-age directive

In this post we will explore `*max-age*` response directive.

> *The `max-age` response directive indicates that the response is to be considered stale after its age is greater than the specified number of seconds.*

What does it mean?

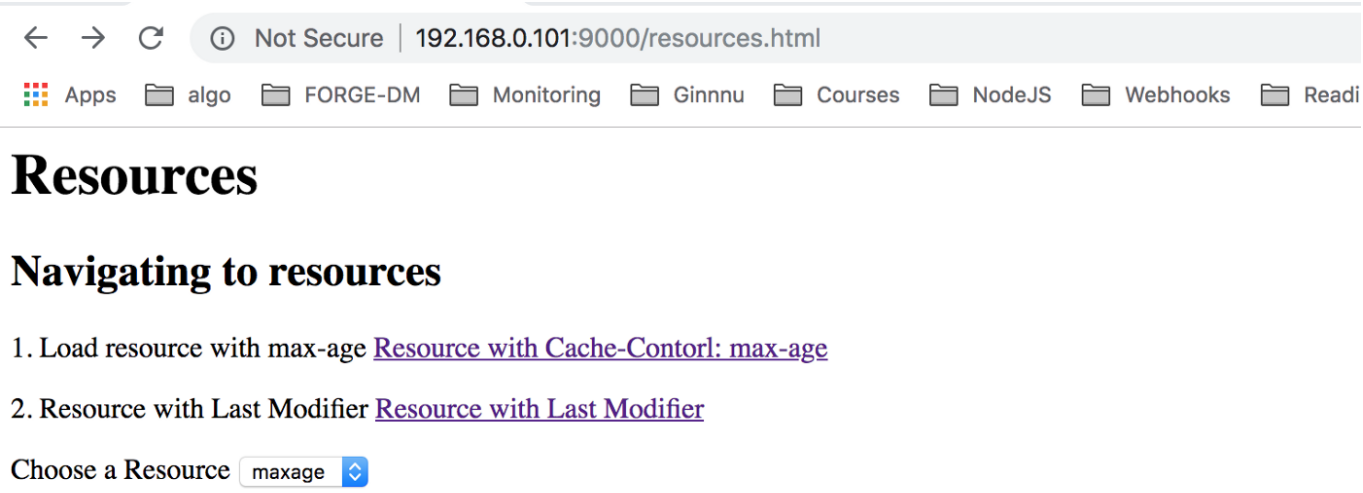
When response for the resource contains `max-age` directive in the header, GET request for the resource should return cached response until expiry (unless the request contains a directive to force revalidation and fetch from origin server.)

Let's explore that via a live server and Chrome browser as client.

All the code for the examples can be found here. Follow the instructions in Readme.MD to build and run the server

Once you have the server running -

- Load the main page by going to `http://<IP|localhost>:9000/resources.html`. It will return a page like below —



Clicking on the link `Resource with Cache-Control:max-age` or choosing `maxage` from the dropdown fetches a response as defined by below method on server

```
{
   response.setHeader('Content-Type', 'text/html');
   response.setHeader('Cache-Control', 'public, max-age=30')
   response.statusCode = 200
   response.end("The awesome resource with max age TTL")
}
```

The server is returning simple text as response but with a directive that *response can be cached for 30 seconds* after its received and its `public` i.e it can be cached *by the browser or any intermediary server* in the response chain.

Lets take a look at the request and response headers -

```
Request Headers (truncated for brevity)

GET /maxage HTTP/1.1
Connection: keep-alive
User-Agent: Mozilla/5.0....
Accept: text/html...
Referer: http://192.168.0.101:9000/resources.html
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

Response headers

HTTP/1.1 200 OK
Content-Type: text/html
Cache-Control: public, max-age=30
Date: Fri, 19 Oct 2018 10:41:33 GMT
Connection: keep-alive
Content-Length: 37

Response
The awesome resource with max age TTL
```

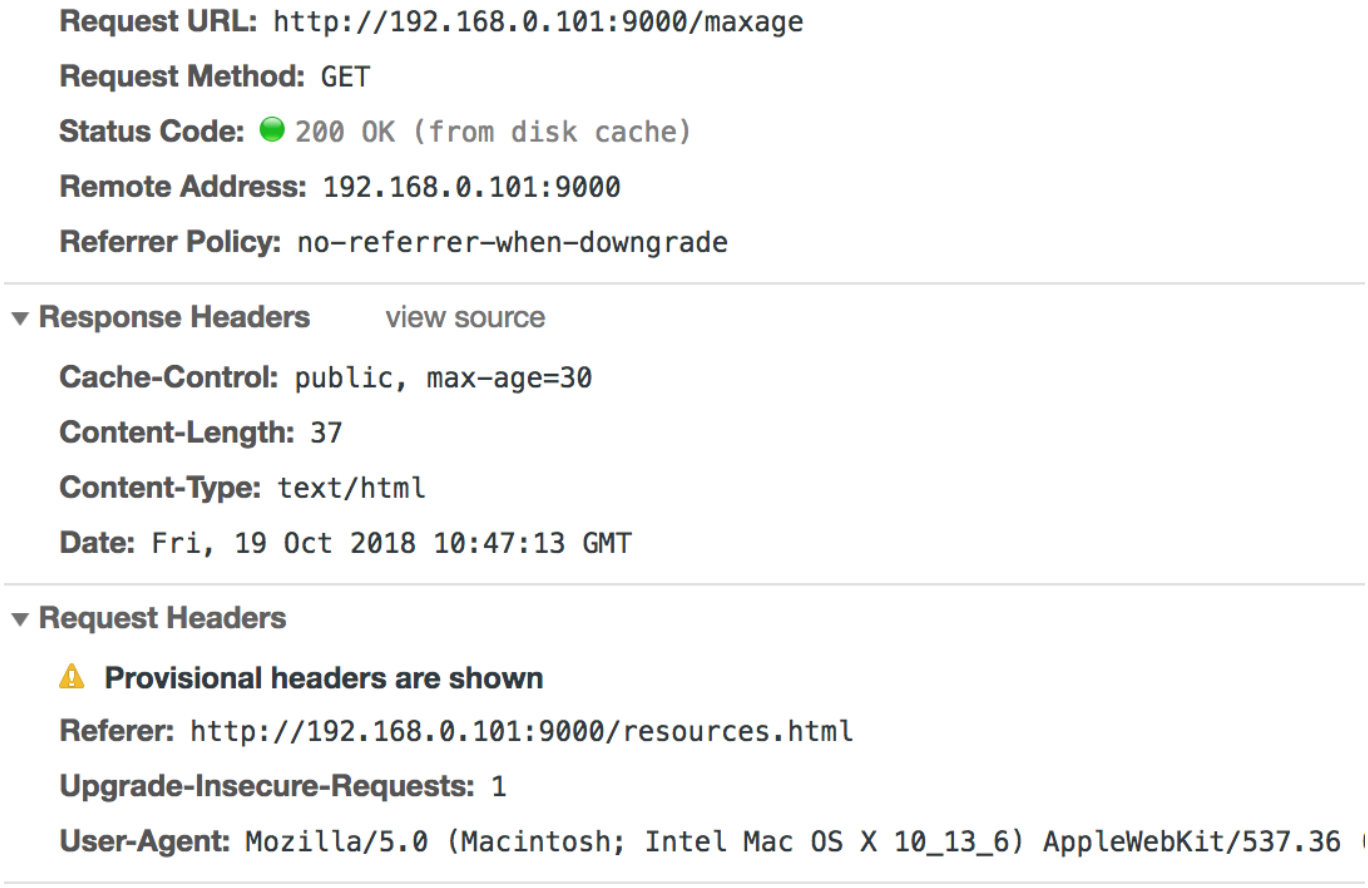
Click the browser back button and then

- Either navigate to the `/maxage` resource via the link on the `resources.html` page or
- Select `maxage` in the dropdown which sends an `xhr` request for `/maxage` resource and shows the response in the same page.

Notice that in either case, response now gets served from the browser `disk cache`.

Here is how the request/response headers look

▼ General



`Status Code: 200 OK (from disk cache)` shows that the response was indeed served from cache. Browser will serve this response from cache until the response expires i.e. in this case, if it has been in cache for more than 30 seconds.

. . .

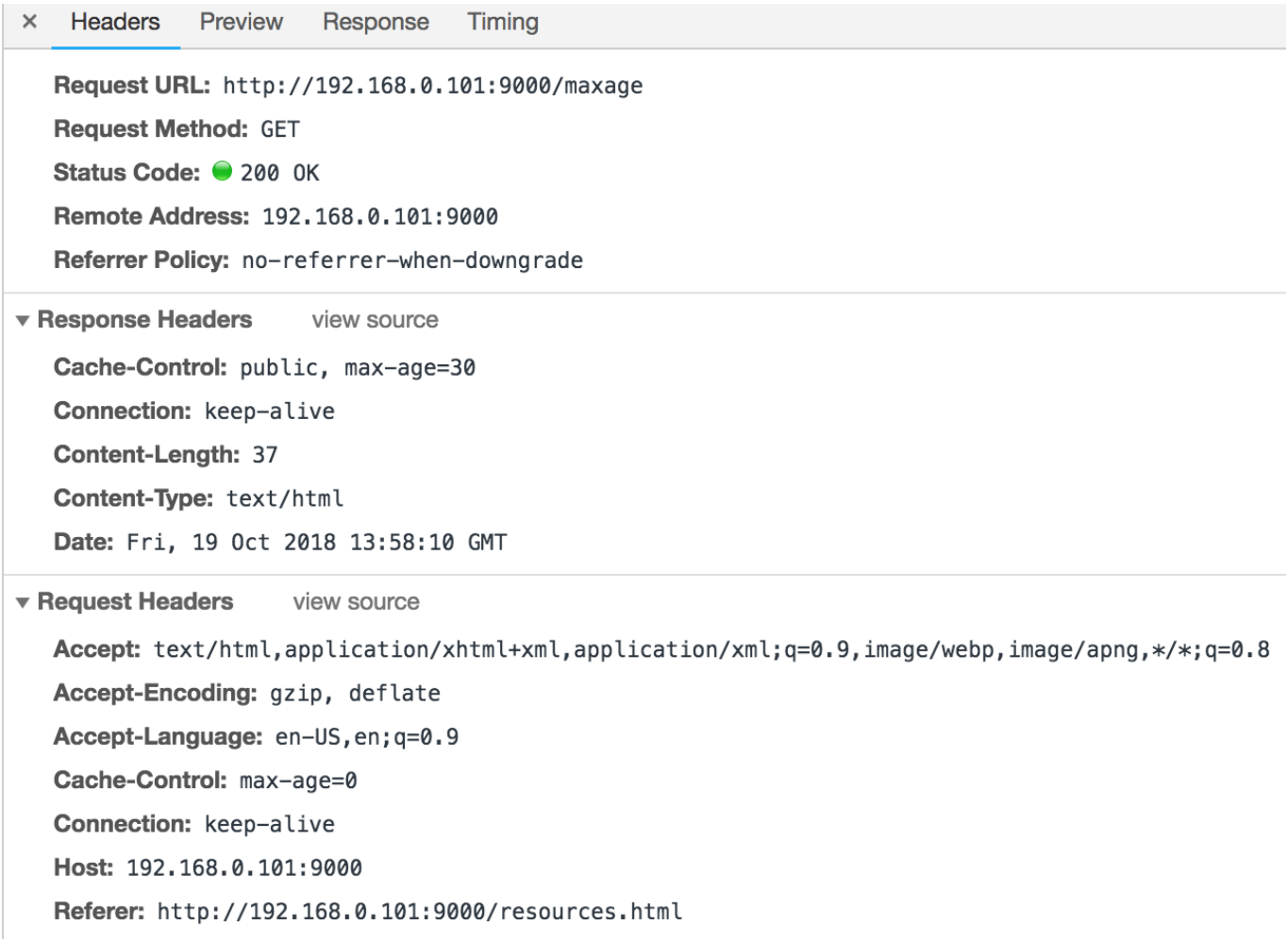
Explicit Cache Invalidation

On the page `http://<IP>:9000/maxage` , if you do a simple refresh (*cmd+r*) or full refresh (*cmd+shift+r*), you will see that the request no longer gets served from disk cache (even within the 30 seconds window)

Why?

- Because Chrome adds `Cache-Control: max-age=0` directive to the resource request on refresh.

See the request headers in Chrome network tab when you refresh —



And as per RFC

> *In particular, a response with either "max-age=0, must-revalidate" or "s-maxage=0" cannot be used to satisfy a subsequent request without revalidating it on the origin server.*

This is called `Specific end to end revalidation`. When the request includes a `max-age=0` cache-control directive, it forces each cache along the path to the origin server to revalidate its own entry, if any, with the next cache or server.

Side Note : It may be debatable to force revalidation even on simple refresh, perhaps using the cache in that case would be more performant but the browser behavior is still in conformance with the HTTP spec

In the **next** part of this series, we will look at `Last-Modified` header, another common way to take advantage of HTTP caching

Thanks for reading !

Web Development Http Cache Browser Cache Cache Control