Client Side Caching

Regular pages

CorePublish sends custom cache/expire headers to the client. Different headers are sent depending on the situation:

Anonymous users

CorePublish sends anonymous users(none-loggedin) users an expire header 1 minute ahead in time, allowing the client to cache the current page for 1 minute. This will cause a much quicker browsing experience on the site, since the browser does not have to send http-if-modified-since headers when navigating back and forward. On the other hand the expire period is short enough to allow editors to change contents without users getting old contents. (you will at longest have to wait 1 minute before all users have the new contents).

Logged in users

Logged-in users however must re-validate every page view since they are actually logged in and the contents of the page can be personal(ized) and dynamic. This applies both to protected and public pages. These users will get an expire header set in the past to force re-validation of each page.

The reason for always re-validating logged in users are: Think of what happens if a logged in user is in on a logged-in page, logs out and then hits “back”. In this case the user should *not* get the content he just saw, since he is now logged in. This means you cannot send expires header in the future when the user is logged in.

Vary: cookie

Another scenario that must be handled is what happens when an anonymous user is on a public page (with expires header in the future) which contains a dynamic tile, e.g. the login form tile.

The user logs in and then goes back to the same page. Since the previous page was public, the user will (for 1 minute) still have a cached version in the browser cache, and the browser will probably display the cached version, which contains a login form and indicates to the user that he is still not logged in (which he is).

This case is solved by adding “Cookie” to the Vary header. The “loggedin” cookie changes when logging in, and thus invalidating the client cache since Vary:cookie tells the client to clear its cache when a cookie value changes.

Pages cached by full cache.

Pages full cached by CtPageCache sends their own headers, but they follow the same rules as above, meaning 1 minute ahead in time unless the cached pages has specified something else in the cache metadata.

> **Note**
>
> Full cached pages are only served to non-loggedin users.

Which classes handles the expire headers?

The expire headers are sent from `CorePublish::bootstrapFrontendMode()`, `CtPageCache::outPutCacheFile()` and `MultimediaArchive::handleGetfile()`.

The HTTP_IF_MODIFIED_SINCE header

The full page cache system and the multimedia archive considers this header and returns a 304 http status header if the requested resource hasn’t changed. Regular, uncached page requests does not support this. The reason for this is that while a cached file knows when in expires, and a single file knows when it was changed, there is no way an uncached page can tell whether some of its (dynamically generated) contents has changed since the last time the client requested that page.

XHR services

Services extending the CpXMLHttpRequest class are normally served as regular pages with the same caching/expire rules as for regular pages. You can however override the expire header sent by implementing the `CpXMLHttpRequest::getClientCacheExpireTimestamp()` and returning another timestamp.

Multimedia files

Files served through getfile.php from the Multimedia archive is given an expire header 5 minutes ahead in time, allowing the client to cache the requested file for 5 minutes. This will cause a much quicker browsing experience on the site, since the browser does not have to send http-if-modified-since headers when navigating back and forward and requesting the files over and over

again. On the other hand the expire period is short enough to allow editors to change contents without users getting old contents. (you will at longest have to wait 5 minutes before all users get the new version of a file, if it is replaced).

The reason for not setting the expire header further ahead in time, is to allow editors to change file contents and make sure that the updated files/images are visible for all clients within reasonable time (in this case, 5 minutes). If we set the expire header e.g. 14 days ahead in time, clients will never get any new version if the editor changes the file.

The expire headers are sent from `MultimediaArchive::handleGetfile()`.