HTTP Caching

By Dave Gash
Dave is a Tech Writer

When someone visits a website, everything the site needs to display and operate has to come from somewhere. All the text, images, CSS styles, scripts, media files, and so on must be retrieved by the browser for display or execution. You can give the browser choices about where it can retrieve a resource from, and that can make a big difference in your page's load speed.

The first time a browser loads a web page, it stores the page resources in the HTTP Cache. The next time the browser hits that page, it can look in the cache for resources that were previously fetched and retrieve them from disk, often faster than it can download them from the network.

While HTTP caching is standardized per the Internet Engineering Task Force (IETF) specifications, browsers may have multiple caches that differ in how they acquire, store, and retain content. You can read about how these caches vary in this excellent article, A Tale of Four Caches.

Of course, every first-time visitor to your page arrives with nothing yet cached for that page. Even repeat visitors may not have much in the HTTP cache; they might have manually cleared it, or set their browser to do so automatically, or forced a fresh page load with a control-key combination. Still, a significant number of your users may revisit your site with at least some of its components already cached, and that can make a huge difference in load time. Maximizing cache usage is critical to speeding up return visits.

Enabling Caching

Caching works by categorizing certain page resources in terms of how frequently or infrequently they change. Your site's logo image, for example, might almost never change, but your site's scripts might change every few days. It's beneficial to you and your users to determine which types of content are more static and which are more dynamic.

It's also important to remember that what we think of as browser caching may in fact take place at any intermediate stop between the original server and the client-side browser, such as a proxy cache or a content delivery network (CDN) cache.

Cache Headers

Two main types of cache headers, cache-control and expires, define the caching characteristics for your resources. Typically, cache-control is considered a more modern and flexible approach than expires, but both headers can be used simultaneously.

Cache headers are applied to resources at the server level – for example, in the `.htaccess` file on an Apache server, used by nearly half of all active websites – to set their caching characteristics. Caching is enabled by identifying a resource or type of resource, such as images or CSS files, and then specifying headers for the resource(s) with the desired caching options.

Cache-control

You can enable cache-control with a variety of options in a comma-delimited list. Here is an example of an Apache `.htaccess` configuration that sets caching for various image file types, as matched by an extension list, to one month and public access (some available options are discussed below).

```
<filesMatch ".(ico|jpg|jpeg|png|gif)$">
 Header set Cache-Control "max-age=2592000, public"
</filesMatch>
```

This example sets caching for styles and scripts, resources that are probably more likely to change than the images, to one day and public access.

```
<filesMatch ".(css|js)$">
 Header set Cache-Control "max-age=86400, public"
</filesMatch>
```

Cache-control has a number of options, often called *directives*, that can be set to specifically determine how cache requests are handled. Some common directives are explained below; you can find more information at the Performance Optimization section and at the Mozilla Developer Network.

- **no-cache**: Somewhat of a misnomer, specifies that content can be cached but, if so, it must be re-validated on each request before being served to a client. This forces the client to check for freshness but allows it to avoid downloading the resource again if it has not changed. Mutually exclusive with **no-store**.
- **no-store**: Indicates that the content actually cannot be cached in any way by any primary or intermediate cache. This is a good option for resources that may contain

sensitive data, or for resources that will almost certainly change from visit to visit. Mutually exclusive with **no-cache**.

- **public**: Indicates that the content can be cached by the browser and by any intermediate caches. Overrides the default **private** setting for requests that use HTTP authentication. Mutually exclusive with **private**.
- **private**: Designates content that may be stored by the user's browser, but may not be cached by any intermediate caches. Often used for user-specific, but not particularly sensitive, data. Mutually exclusive with **public**.
- **max-age**: Defines the maximum time that the content may be cached before it must be revalidated or downloaded again from the original server. This option generally replaces the expires header (see below) and takes a value in seconds, with a maximum valid age of one year (31536000 seconds).

Expires Caching

You can also enable caching by specifying expiration, or expiry, times for certain types of files, which tell browsers how long to use a cached resource before requesting a fresh copy from the server. The expires header just sets a time in the future when the content should expire. After that point, requests for the content must go back to the original server. With the newer and more flexible cache-control header, the expires header is often used as a fallback.

Here's an example of how you might set up caching in the `.htaccess` file on an Apache server.

```
## EXPIRES CACHING ##
ExpiresActive On
ExpiresByType image/jpg "access plus 1 year"
ExpiresByType image/jpeg "access plus 1 year"
ExpiresByType image/gif "access plus 1 year"
ExpiresByType image/png "access plus 1 year"
ExpiresByType text/css "access plus 1 month"
ExpiresByType application/pdf "access plus 1 month"
ExpiresByType text/x-javascript "access plus 1 month"
ExpiresByType application/x-shockwave-flash "access plus 1 month"
ExpiresByType image/x-icon "access plus 1 year"
ExpiresDefault "access plus 2 days"
## EXPIRES CACHING ##
```

(source: GTmetrix)

As you can see, different types of files have different expiry dates in this example: images don't expire for a year after access/caching, while scripts, PDFs, and CSS styles expire in a

month, and any file type not explicitly listed expires in two days. The retention periods are up to you, and should be chosen based on the file types and their update frequency. For example, if you regularly change your CSS, you might want to use a shorter expiry, or even none at all, and let it default to the two-day minimum. Conversely, if you link to some static PDF forms that almost never change, you might want to use a longer expiry for them.

Tip: Don't use an expiry greater than one year; that's effectively forever on the internet and, as noted above, is the maximum value for `max-age` under cache-control.

Summary

Caching is a reliable and low-hassle way to improve your pages' load speed and thus your users' experience. It is powerful enough to allow sophisticated nuances for specific content types, but flexible enough to allow easy updates when your site's content changes.

Be assertive with caching, but also be aware that if you later change a resource that has a long retention period, you may inadvertently deprive some repeat visitors of newer content. You can find a great discussion of caching patterns, options, and potential pitfalls in Caching Best Practices and Max-age Gotchas.

Last updated July 2, 2018.