Programmable Cache-Control: One Size Does Not Fit All

Lori MacVittie, 2013-25-03

#webperf For addressing challenges related to performance of #mobile devices and networks, caching is making a comeback.

It's interesting - and almost amusing - to watch the circle of technology run around best practices with respect to performance over time. Back in the day caching was the ultimate means by which web application performance was improved and there was no lack of solutions and techniques that manipulated caching capabilities to achieve optimal performance.

Then it was suddenly in vogue to address the performance issues associated with Javascript on the client. As Web 2.0 ascended and AJAX-based architectures ruled the day, Javascript was Enemy #1 of performance (and security, for that matter). Solutions and best practices began to arise to address when Javascript loaded, from where, and whether or not it was even active.

And now, once again, we're back at the beginning with caching. In the interim years, it turns out developers have not become better about how they mark content for caching and with the proliferation of access from mobile devices over sometimes constrained networks, it's once again come to the attention of operations (who are ultimately responsible for some reason for performance of web applications) that caching can dramatically improve the performance of web applications.

[Excuse me while I take a breather - that was one long thought to type.]

Steve Souders, web performance engineer extraordinaire, gave a great presentation at HTML5DevCon that was picked up by High Scalability: Cache is King!.

The aforementioned articles notes:

> **Use HTTP cache control mechanisms**: max-age, etag, last-modified, if-modified-since, if-none-match, no-cache, must-revalidate, no-store. Want to prevent HTTP sending conditional GET requests, especially over high latency mobile networks. Use a long max-age and change resource names any time the content changes so that it won't be cached improperly.
>
> -- Better Browser Caching Is More Important Than No Javascript Or Fast Networks For HTTP Performance

The problem is, of course, that developers aren't putting all these nifty-neato-keen tags and meta-data in their content and the cost to modify existing applications to do so may result in a prioritization somewhere right below having an optional, unnecessary root canal. In other cases the way in which web applications are built today - we're still using AJAX-based, real-time updates of chunks of content rather than whole pages - means simply adding tags and meta-data to the HTML isn't necessarily going to help because it refers to the *page* and not the data/content being retrieved and updated for that "I'm a live, real-time application" feel that everyone has to have today.

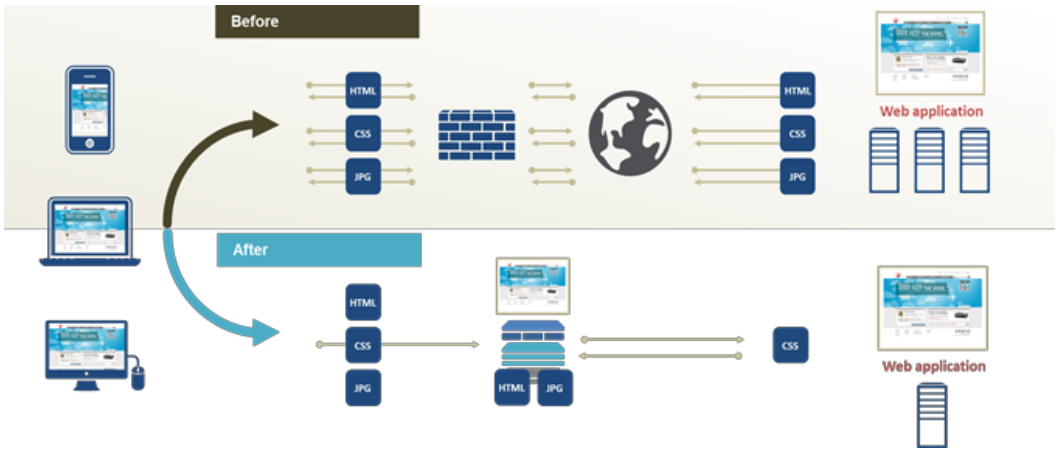
Too, caching tags and meta-data in HTML doesn't address every type of data. JSON, for example, commonly returned as the response to an API call (used as the building blocks for web applications more and more frequently these days) aren't going to be impacted by the HTML caching directives. That has to be addressed in a different way, either on the server side (think Apache mod_expire) or on the client (HTML5 contains new capabilities specifically for this purpose and there are usually cache directives hidden in AJAX frameworks like jQuery).

The Programmable Network to the Rescue

What you need is the ability to insert the appropriate tags, on the appropriate content, in such a way as to make sure whatever you're about to do (a) doesn't break the application and (b) is actually going to improve the performance of the end-user experience for that specific request.

Note that (b) is pretty important, actually, because there are things you do to content being delivered to end users on mobile devices over mobile networks that might make things worse if you do it to the same content being delivered to the same end user on the same device over the wireless LAN. Network capabilities matter, so it's important to remember that.

To avoid rewriting applications (and perhaps changing the entire server-side architecture by adding on modules) you could just take advantage of programmability in the network. When enabled as part of a full-proxy, network intermediary the ability to programmatically modify content in-flight becomes invaluable as a mechanism for improving performance, particularly with respect to adding (or modifying) cache headers, tags, and meta-data.



By allowing the intermediary to cache the cacheable content while simultaneously inserting the appropriate cache control headers to manage the client-side cache, performance is improved. By leveraging programmability, you can start to apply device or network or application (or any combination thereof) logic to manipulate the cache as necessary while also using additional performance-enhancing techniques like compression (when appropriate) or image optimization (for mobile devices).

The thing is that a generic "all on" or "all off" for caching isn't going to always result in the best performance. There's *logic* to it that says you need the capability to say "*if X and Y then ON else if Z then OFF*". That's the power of a programmable network, of the ability to write the kind of logic that takes into consideration the context of a request and takes the appropriate actions in real-time.

Because one size (setting) simply does not fit all.

F5 Networks, Inc. | 401 Elliot Avenue West, Seattle, WA 98119 | 888-882-4447 | f5.com

F5 Networks, Inc.
Corporate Headquarters
info@f5.com

F5 Networks
Asia-Pacific
apacinfo@f5.com

F5 Networks Ltd.
Europe/Middle-East/Africa
emeainfo@f5.com

F5 Networks
Japan K.K.
f5j-info@f5.com