Bien gérer le cache navigateur pour augmenter les performances d'un site

Par Aymeric Bouillat

Domaine :	Recherche	**Référencement**
Niveau :	Pour tous	**Avancé**

Dans le cadre de l'optimisation de l'accélération du temps d'affichage de vos pages web, pour le plus grand bénéfice de Google mais surtout de l'internaute, il existe une large gamme de possibilités à mettre en œuvre. L'une d'elles est la bonne configuration des en-tête http transmises par le serveur afin de gérer au mieux le cache du navigateur distant. Voici quelques informations et astuces pour mettre en place cela au mieux.

Nous l'avons vu dans la lettre Recherche et Référencement du mois dernier (http://recherche-referencement.abondance.com/2015/09/temps-de-chargement-et-seo-ce-quil-faut.html), le temps de chargement d'une page Web a un impact indirect sur votre référencement (optimisation du crawl de Googlebot, pogosticking, etc.) mais également de façon directe comme le brevet Google « *Using resource load times in ranking search results* » (http://www.google.com/patents/US8645362) nous le montre.

La récupération de l'ensemble des fichiers qui composent une page Web (images, CSS, JS) peut s'avérer coûteuse en terme de temps. L'appel de chacun des éléments qui la compose provoque de nombreux allers-retours entre le serveur du site web et le navigateur de l'internaute : plus le serveur doit gérer de requêtes, plus ses temps de réponses risqueront d'être élevés, et plus les requêtes seront nombreuses, plus l'affichage d'une page Web complète sera retardée par le navigateur.

La mise en cache par le navigateur permettra donc de servir rapidement des fichiers déjà récupérés, en limitant le nombre de requêtes vers le serveur d'un site Web. Cela sera au final bénéfique pour l'expérience utilisateur, et indirectement donc pour le référencement, car la charge serveur sera allégée et les temps de réponses seront plus performants pour servir les éléments de chaque page HTML.

Le Cache Navigateur et les en-têtes de cache

Toutes les informations concernant les directives de cache se trouvent dans les en-têtes http. Plusieurs outils vous permettront de voir ces en-têtes HTTP pour vérifier leur bonne configuration, certains sont disponibles en ligne comme https://redbot.org/ ou d'autres sous la forme de plugins Chrome/Firefox comme HTTP Headers (https://addons.mozilla.org/fr/firefox/addon/live-http-headers/). Mais, afin de diagnostiquer rapidement une mauvaise gestion du cache navigateur, Page Speed Insights de Google sera très utile (figure 1).

Dans l'exemple de la figure 1, les feuilles de styles seront téléchargées à nouveau entre chaque page lors de la navigation sur le site, qui se révèlera donc moins fluide que ce qu'elle pourrait être. La mise en cache côté client permettra de limiter le nombre de requêtes vers le serveur ainsi que la bande passante vers ce dernier.

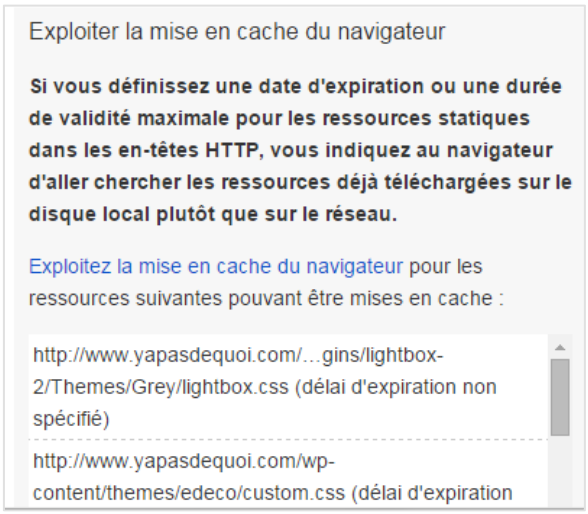

Fig.1. Problème de mise en cache détecté par l'outil Google Page Speed Insights.

En-têtes de cache, oui mais lesquels ?

Différents en-têtes de cache existent. Certaines sont apparues avec les évolutions de protocole HTTP/1.0 → HTTP/1.1) et permettent de savoir comment distribuer le cache de façon efficace. Nous allons détailler chacun d'entre eux, avec ses avantages et inconvénients

Expires : protocole HTTP/1.0, expiration

L'en-tête *Expires* envoyé pour un fichier donné définit une date future à laquelle le fichier ne sera plus valable, et devra être re-téléchargé sur le serveur. Ex : http://monsite.com/exemple.png
Expires: Thu, 04 Jun 2016 20:49:14 GMT

Une fois dans le cache, le fichier reste valable jusqu'au 04 Juin 2016 à 20h49 et 14 secondes (fig.2).

L'avantage de cet en-tête est qu'il ne déclenche aucune requête intermédiaire jusqu'à la date d'expiration. Le fichier statique sera servi à partir du cache navigateur sans aucune sollicitation du serveur. Mais il présente malgré tout quelques inconvénients dans ce cas de figure :
• La modification du fichier original ne mettra pas à jour le cache pendant un an ;
• La date d'expiration envoyée lors de la 1ère demande dépend de l'heure du serveur (celle-ci doit être correctement paramétrée).

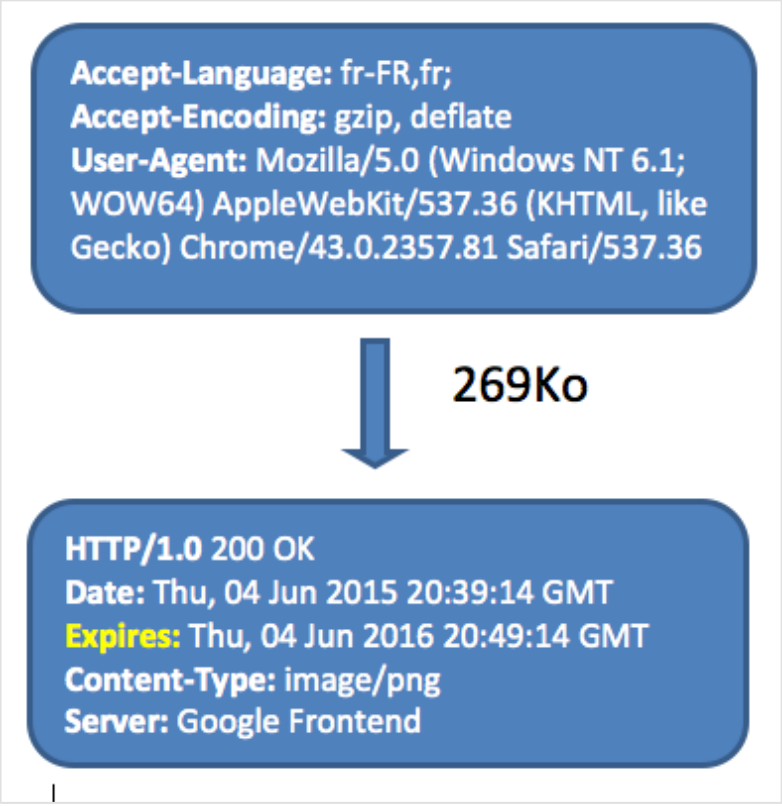

Fig. 2. L'en-tête Expires.

Par ailleurs, si la date est la même pour tous les internautes, cela pourrait provoquer un pic de charge le jour de l'expiration, si plusieurs fichiers ont la même date de fin. Heureusement, Apache permet de maitriser plus finement cette en-tête Expires via son module *mod_expire*s, en fonction de la date d'accès au fichier par l'internaute :

ExpiresByType image/png "access plus 1 month"

Chaque navigateur conservera l'image dans son cache dans un délai d'un mois à compter de la date à laquelle le fichier a été récupéré. Cet en-tête est utile pour les fichiers statiques de type images qui changent peu, ce qui n'est pas nécessairement le cas des JS/CSS qui pourraient compromettre le bon affichage du site, en cas de modification de l'un d'entre eux.

Last-modified : protocole HTTP/1.1, validation

L'en-tête *Last-modified* envoyé pour un fichier donné défini la date à laquelle le fichier a été modifié pour la dernière fois sur le serveur.
Ex : http://monsite.com/exemple.png
Last-Modified: Thu, 02 Jun 2015 17:32:03

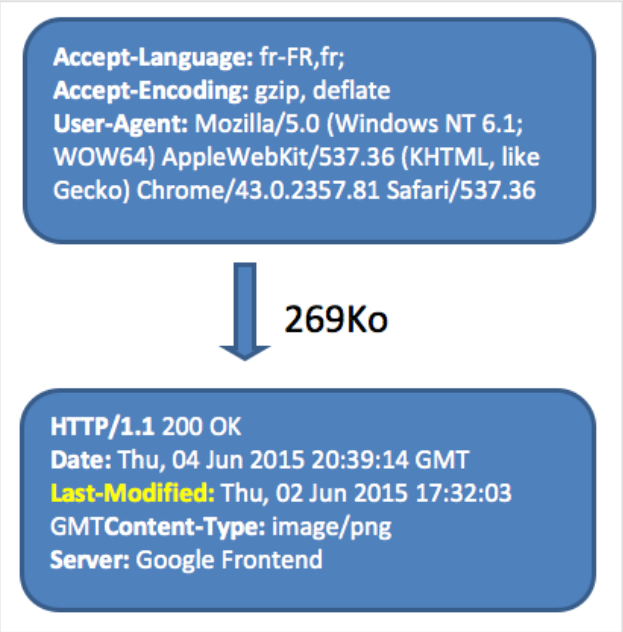

Fig. 3. L'en-tête Last Modified.

A partir du moment où le fichier se trouve dans le cache du navigateur, s'il est appelé par une autre page Web, le navigateur enverra une requête avec un en-tête http spécifique *If-Modified-Since* pour ce fichier, contenant la dernière date de modification connue du fichier.

Dans le cas où le fichier n'a pas été modifié, et que la dernière date de modification sur le serveur correspond à celle de la version en cache, le serveur ne renverra qu'une réponse partielle de type HEAD avec un statut *HTTP 304*, qui indique que le fichier n'a pas été modifié depuis qu'il a été mis en cache pour la première fois.

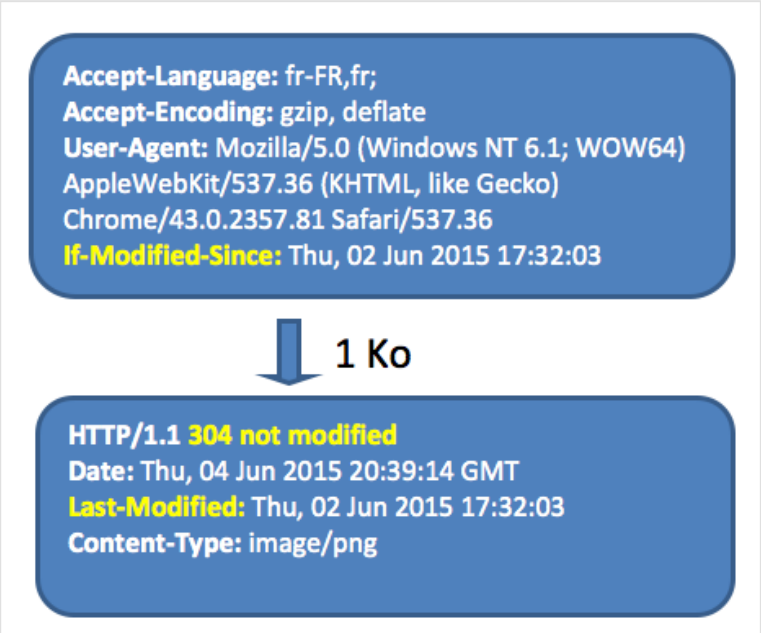

Fig. 4. L'en-tête If-Modified-Since.

A l'inverse, si le fichier a été modifié, le serveur renverra au navigateur le nouveau fichier avec un en-tête *Last-Modified* mis à jour.

Cet en-tête consomme donc une requête supplémentaire afin de valider le fichier, sans forcément le récupérer en intégralité. Elle a l'avantage de permettre une bonne fraîcheur des fichiers dans le cache du navigateur (ex : mise à jour d'un thème et de ses feuilles de styles par exemple)

Etag : protocole HTTP/1.1, validation

L'en-tête Etag (pour Entity Tag) envoyé pour un fichier donné défini une empreinte pour celui-ci.
Ex : http://monsite.com/exemple.png
ETag: "UreJ2g"

Elle fonctionne sur le même principe que *Last-Modified,* mais avec un jeton de validation correspondant à une empreinte unique pour chaque fichier. Si un fichier est modifié, son empreinte sera modifiée et il sera à nouveau téléchargé vers le navigateur.

L'en-tête intermédiaire envoyé est *If-None-Match* qui envoie le jeton connu (UreJ2g dans notre exemple) afin que le serveur puisse faire la comparaison avec l'empreinte du fichier.

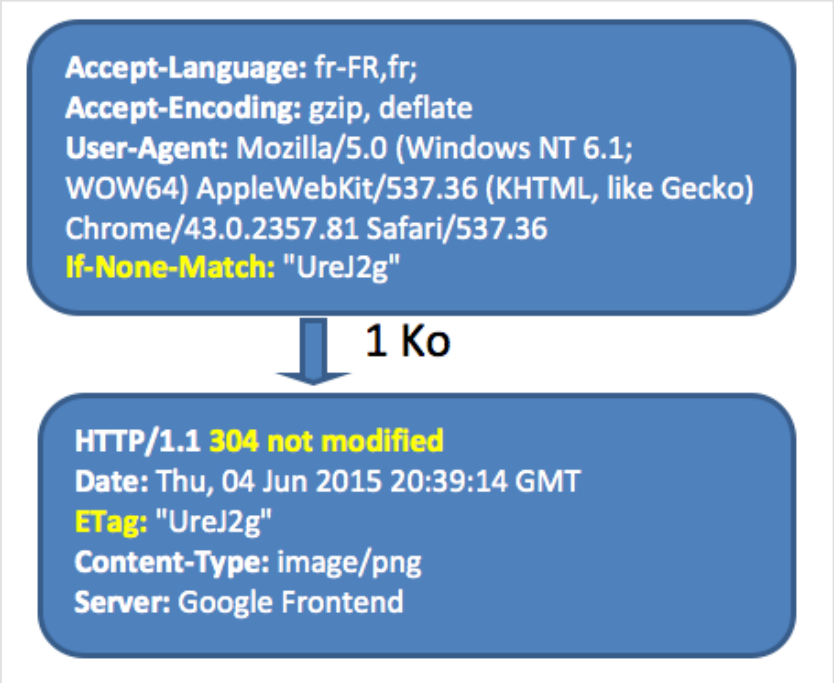

Fig. 5. L'en-tête Etag.

Aucun transfert ne sera effectué tant que l'empreinte du fichier en cache sera identique à celle du fichier sur le serveur. On ne renvoie qu'une réponse partielle (status 304) via une requête intermédiaire.

Cache-Control : protocole HTTP/1.1, validation précise

Cet en-tête remplace l'en-tête *Expires* défini par le protocole HTTP/1.0. Il permet une gestion du cache plus fine pour chaque ressource, en intégrant une gestion du cache des serveurs mandataires ou proxys intermédiaires. Les différentes directives de cet en-tête *Cache-Control* doivent être séparées par des virgules.

Ex. : http://monsite.com/exemple.png
Cache-Control: public; max-age=31536000

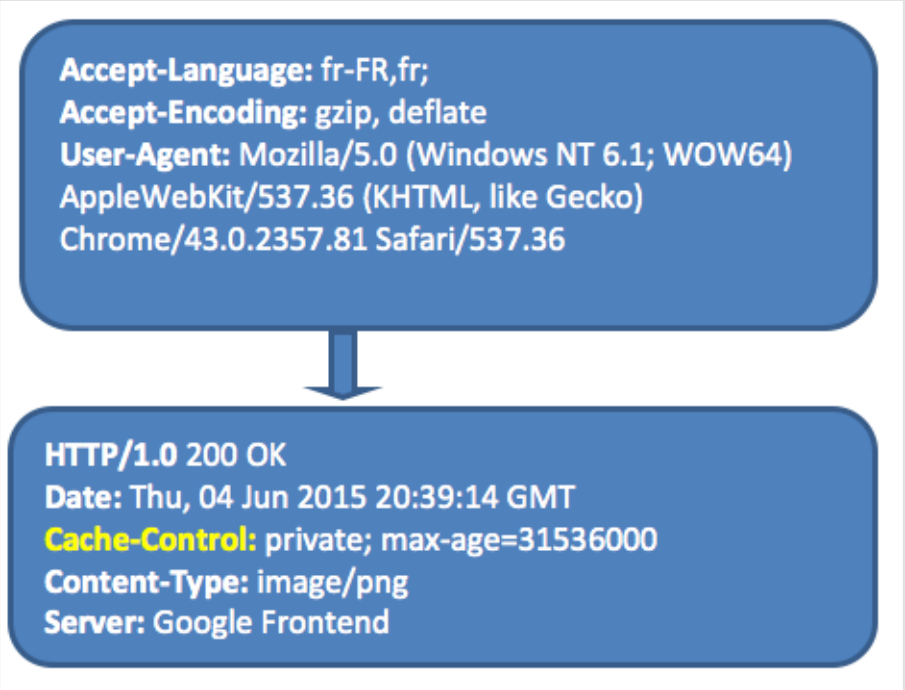

Fig. 6. L'en-tête Cache-Control.

Les différentes directives possibles sont les suivantes :

max-age=31536000 : On indique ici que le fichier est considéré comme frais pour une durée de 31 536 000 secondes, soit 1 an, à partir du moment où il est récupéré depuis le serveur (la date n'est donc pas absolue, à l'inverse d'*Expires*). Il n'est pas rare de voir ce type d'en-tête sur des pages d'accueil de sites Web régulièrement mise à jour : *Cache-Control : max-age=600* (mise en cache de dix minutes)

private : Le fichier ne doit pas être mis en cache par les serveurs mandataires, entre l'internaute et le serveur du site Web (ex : proxy d'entreprise partagé ou encore reverse proxy).

A l'inverse, la directive *public* indique que tous les systèmes de cache (navigateurs, proxys), peuvent mettre en cache la ressource. Ainsi, une ressource authentifiée sera considérée comme publique, à manier avec précaution donc.

La directive *no-cache* indique quant à elle que la ressource demandée ne peut pas être mise en cache, et qu'elle devra être récupérée si l'élément est à nouveau demandé par le serveur de cache ou le navigateur. La combinaison *Cache-Control: public, no-cache* évitera par exemple la mise en cache d'éléments accessibles avec mot de passe.

Bien gérer son cache navigateur

Le fait que ces en-têtes (*Expires, Cache-Control*) peuvent se retrouver en concurrence pour une seule et même ressource peut devenir problématique : en fonction des navigateurs dont l'algorithme de gestion de cache peut varier, *Expires* peut avoir le dessus sur *Cache-Control* et inversement.

Certains CMS renvoient d'eux même des en-têtes de cache de type *Cache-Control*, qui peuvent rentrer en conflit avec des en-têtes *Expires*. Il est indispensable dans ce type de cas de trouver l'origine des en-têtes générés, afin d'avoir un contrôle plus fin sur ces derniers. Des analyseurs de trames comme Fiddler, ou l'inspecteur d'éléments de Chrome/Firefox (onglet *Network*, voir fig.7) vous permettront de mieux comprendre la façon dont est géré le cache pour chacune de vos ressources.

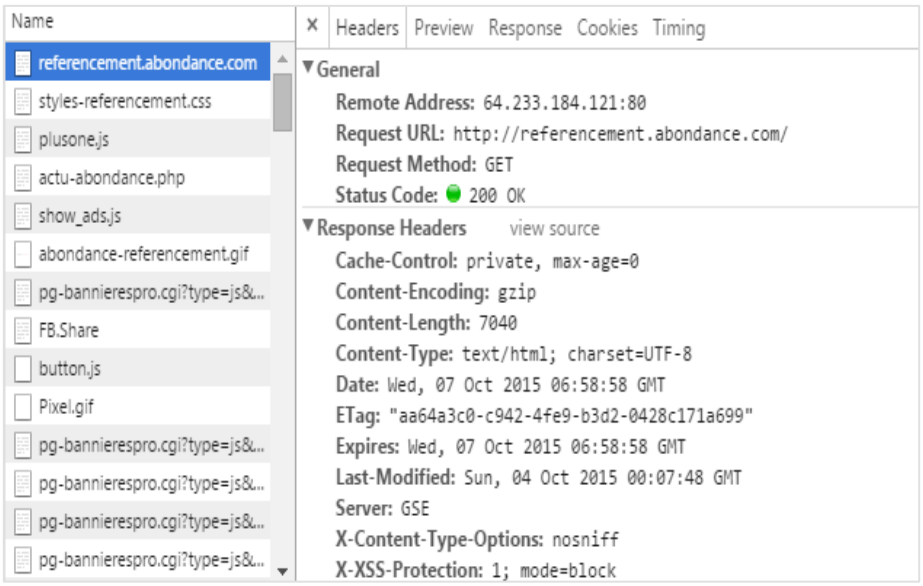

Fig. 7. En-têtes HTTP dans l'inspecteur d'éléments de Chrome

L'utilisation du Versioning reste une solution souvent utilisée pour forcer la mise à jour de fichiers, malgré les en-têtes de cache : on renomme l'URL d'un élément en y ajoutant un identifiant de version dans les paramètres d'URL, ce qui forcera le téléchargement d'une nouvelle ressource quand la page HTML faisant référence à ces éléments sera appelée.
Ex : */styles/ma-feuille-de-style.css?**v=0.42***

On voit encore trop souvent des sites pour lesquels la gestion des fichiers statiques n'est pas paramétrée et sont récupérés en intégralité entre chaque page web. De la même manière, la gestion des allers-retours sur un même site web plusieurs fois dans la journée peut s'avérer coûteuse en termes de ressources sur le serveur et provoquer une hausse des temps de chargement, si aucun cache n'est défini, aussi court soit-il.

Pour une meilleure expérience utilisateur et afin d'améliorer les performances de votre site, il est donc indispensable de gérer correctemment le cache pour chaque type de ressource qu'un site Web peut servir : utilisez page Speed Insights pour diagnostiquer rapidement ce type d'erreur. En parallèle, les modules *mod_expires* ou encore *mod_headers* d'Apache vous permettront de contrôler au mieux ces en-têtes de cache, en cas de besoin. Gardez le en tête !

Aymeric Bouillat, *consultant SEO, Resoneo (http://twitter.com/aymerictwit ou http://www.yapasdequoi.com)*