Un tutoriel de la mise en cache

pour les auteurs Web et les webmestres

Ce document est informatif. Bien que de nature technique, il essaye de rendre les concepts mis en jeu compréhensibles et applicables à des situations concrètes. Pour cette raison, certains aspects de la documentation sont simplifiés ou omis, par souci de clarté. Si votre intérêt sur le sujet vous porte aux détails, veuillez explorer le chapitre « Références et autres informations » à la fin.

Qu'est-ce qu'un cache Web ? À quoi servent-ils ?

Un *cache Web* se tient entre un ou plusieurs serveurs Web (appelés aussi *serveurs originaux*) et un ou plusieurs clients, et il observe le va-et-vient des requêtes en enregistrant pour lui-même des copies des réponses — comme des pages HTML, des images et des fichiers (appelés collectivement des *représentations*). Par la suite, s'il se présente une autre requête pour la même adresse URL, il pourra utiliser la réponse qu'il détient au lieu de la réclamer à nouveau au serveur original.

On utilise des caches Web pour deux raisons principales :

- Pour **réduire le temps d'attente** — Puisqu'on satisfait la requête à partir du cache (plus proche du client) au lieu du serveur original, l'obtention de la représentation et son affichage prennent moins de temps. Le Web semble plus réactif.
- Pour **réduire le trafic des réseaux** — Puisqu'on réutilise les représentations, la bande passante consommée par le client est réduite. Si le client paye au débit, il économise de l'argent, et il contient ses besoins en bande passante et les gère mieux.

Les types de cache Web

Les caches de navigateurs

Si vous examinez le boîte de dialogue des préférences d'un navigateur Web moderne (comme Internet Explorer, Safari ou Mozilla), vous remarquerez probablement un réglage de « cache ». Il vous permet de consacrer une partie du disque dur de votre ordinateur au stockage des représentations que vous avez vu, juste pour vous. Le cache du navigateur fonctionne selon des règles plutôt simples. Il vérifiera la fraîcheur des représentations, habituellement une fois par session (c'est-à-dire, pour l'invocation actuelle du navigateur).

Ce cache est particulièrement utile lorsque l'utilisateur actionne le bouton « retour » ou clique un lien pour voir une page qu'il vient juste de visiter. De même, si vous utilisez les mêmes images pour la navigation dans tout votre site, elles seront servies depuis le cache du navigateur presque instantanément.

Les caches de serveurs mandataires

Les caches de serveurs mandataires Web fonctionnent selon le même principe mais sur une échelle beaucoup plus vaste. Les mandataires servent des centaines ou des milliers d'utilisateurs de la même façon ; les grandes entreprises et les fournisseurs d'accès Internet les mettent souvent en place sur leurs pare-feux, ou comme dispositifs autonomes (appelés aussi *intermédiaires*).

Puisque les caches de mandataires ne font pas partie du client ou du serveur original, mais se trouvent plutôt ailleurs dans le réseau, les requêtes doivent leur être envoyées d'une façon ou d'une autre. L'un des moyens d'y parvenir est d'utiliser le réglage de mandataire de votre navigateur pour lui indiquer lequel utiliser ; un autre moyen est d'utiliser l'interception. Les requêtes Web sont redirigées aux *serveurs mandataires d'interception* par le réseau sous-jacent même, de sorte qu'il n'est pas nécessaire de configurer les clients pour eux, ou même qu'ils connaissent leur existence.

Les caches de mandataires appartiennent à un type de *cache partagé* ; plutôt que d'avoir un seul utilisateur, ils en ont habituellement un grand nombre et, pour cette raison, ils sont très bons pour réduire l'attente et le trafic des réseaux. Parce que les représentations populaires sont réutilisées plusieurs fois.

Les caches de passerelles

Connus aussi comme « serveurs caches inverses » ou « caches auxilliaires », les caches de passerelles sont également des intermédiaires, mais au lieu d'être déployés par les administrateurs de réseaux pour économiser la bande passante, ils le sont typiquement par les webmestres mêmes, pour rendre leurs sites plus extensibles, fiables, et avec de meilleures performances.

Les requêtes peuvent être routées aux caches de passerelles selon plusieurs méthodes, mais on se sert d'une certaine forme d'équilibrage de charge afin qu'une ou plusieurs d'entre elles passent pour le serveur original auprès des clients.

Des *réseaux de diffusion de contenu* (CDN) distribuent des caches de passerelles dans tout l'Internet (ou dans une partie) et vendent de la mise en cache aux sites Web intéressés. Speedera et Akamai sont des exemples de réseaux de diffusion de contenu.

Ce tutoriel se concentre essentiellement sur les caches de navigateurs et de mandataires, quoiqu'une partie des informations puisse également convenir pour ceux intéressés par les caches de passerelles.

Les caches Web ne sont-ils pas mauvais pour moi ? Pourquoi devrais-je les aider ?

La mise en cache est l'une des technologies les plus méconnues sur l'Internet. Les webmestres notamment craignent de perdre le contrôle de leurs sites, au motif qu'un serveur cache peut les « dissimuler » à leurs utilisateurs, en rendant difficile l'identification de celui qui utilise le site.

Malheureusement pour eux, même si les caches Web n'existaient pas, l'Internet est régi par trop de variables pour garantir l'obtention d'une image exacte de la façon dont les utilisateurs voient leurs sites. Si c'est un gros souci pour vous, ce tutoriel vous apprendra à obtenir les statistiques dont vous avez besoin, sans rendre votre site inamical aux caches.

Un autre souci est celui selon lequel les caches sont susceptibles de servir du contenu périmé, ou *vieux*. Au contraire, ce tutoriel peut vous montrer comment configurer votre serveur afin de contrôler la mise en cache de votre contenu.

Inversement, si vous planifiez bien votre site, les caches peuvent aider au chargement plus rapide de votre site, et limiter la charge sur votre serveur et votre lien Internet. La différence peut être considérable : un site difficile à mettre en cache peut prendre plusieurs secondes à charger tandis qu'un autre exploitant avantageusement la mise en cache semblera peut-être instantané en comparaison. Les utilisateurs apprécieront un site à chargement rapide et le visiteront plus souvent.

> Les réseaux de diffusion de contenu (CDN) constituent un développement intéressant car à la différence de beaucoup de serveurs caches, leurs passerelles caches s'accordent aux intérêts des sites Web mis en cache, de sorte que ces problèmes n'ont pas lieu. Par contre, même si vous utilisez un réseau CDN, vous devez toujours tenir compte du fait qu'il y aura des serveurs caches et des caches de navigateurs en aval.

Voyez-le sous cet angle : beaucoup de grandes compagnies de l'Internet dépensent des millions de dollars à installer des grappes de serveurs répartis dans le monde pour copier leur contenu, afin d'en rendre l'accès aussi rapide que possible pour leurs utilisateurs. Les caches font la même chose pour vous et ils se trouvent encore plus près de l'utilisateur final. Et cerise sur le gateau, vous n'avez rien à payer.

Le fait est que des caches de mandataires et de navigateurs seront utilisés, que ça vous plaise ou non. Si vous ne configurez pas votre site pour sa bonne mise en cache, il le sera selon les paramètres par défaut qu'aura décidé l'administrateur du cache.

Comment fonctionnent les caches Web

Tous les caches possèdent un jeu de règles qu'ils utilisent pour déterminer quand servir une représentation du cache, si celle-ci est disponible. Certaines de ces règles sont fixées dans les protocoles (HTTP 1.0 et 1.1), et d'autres par l'administrateur du cache (que ce soit l'utilisateur du cache de navigateur, ou l'administrateur du mandataire).

Généralement parlant, il s'agit des règles suivies le plus couramment (ne vous inquiétez pas si les détails vous échappent, l'explication suit) :

1. Si les en-têtes de la réponse disent au cache de ne pas la garder, il ne le fait pas ;
2. Si aucun validateur (une en-tête `ETag`, ou `Last-Modified`) n'est présent dans la réponse, elle sera considérée comme ne pouvant être caché ;
3. Si la requête est authentifiée ou sécurisée, elle n'est pas mise en cache.
4. Une représentation en cache est censée être *fraîche* (c'est-à-dire, prête à être envoyée à un client sans vérification auprès du serveur original) si :
   - Aucune en-tête de date d'expiration ou autre contrôle de date ne lui a été attribuée, et elle est toujours dans la période de fraîcheur ;
   - Le cache du navigateur, réglé pour une vérification une fois par session, a déjà vu la représentation ;
   - Un cache de mandataire a vu la représentation récemment, celle-ci ayant été modifié il y a relativement longtemps.

   Les représentations fraîches sont servies directement depuis cache, sans vérification auprès du serveur original.
5. Si une représentation est vieille, le serveur original sera interrogé afin de la *valider*, ou de dire au cache si la copie est toujours bonne ou pas.

La *fraîcheur* et la *validation* sont les moyens les plus importants du cache pour travailler avec le contenu. Une représentation fraîche sera instantanément disponible dans le cache, tandis qu'une représentation validée évitera de l'envoyer à nouveau entièrement si elle n'a pas changé.

Comment contrôler les caches (ou ne pas les contrôler)

Les concepteurs Web et les webmestres disposent de plusieurs outils pour afiner le traitement de leurs sites par les caches. Il faudra peut-être mettre les mains dans la configuration de votre serveur, mais le jeu en vaut la chandelle. Pour des précisions à propos de l'utilisation de ces outils avec votre serveur, cf. les sections ci-après concernant la mise en œuvre.

Les balises Meta HTML et les en-têtes HTTP

Les auteurs HTML peuvent placer, dans la section <HEAD> d'un document, des balises qui en décrivent les attributs. On utilise souvent ces *balises meta* dans la croyance qu'elles peuvent marquer le document comme non cachable, ou le faire expirer à une date donnée.

Les balises meta sont faciles à utiliser mais ne sont pas très efficaces. Car elles ne sont respectées que par quelques caches de navigateurs (qui lisent réellement le code HTML), pas par les caches mandataires (qui ne lisent quasiment jamais le code HTML dans le document). Quoiqu'on puisse être tenté de placer une balise meta « `Pragma: no-cache` » dans une page Web, elle ne sera pas toujours tenue fraîche pour autant.

En effet, les *en-têtes HTTP* véritables vous donnent un grand pouvoir sur la façon dont les caches de navigateurs et les mandataires gèrent vos représentations. Elles sont invisibles dans le code HTML et sont habituellement générée par le serveur Web. Toutefois, ce contrôle s'exerce à un certain degré, selon le serveur utilisé. Dans les sections suivantes, vous verrez quelles sont les en-têtes HTTP intéressantes et comment les appliquer à votre site.

> Si votre site est hébergé chez un fournisseur d'accès Internet, ou dans une grappe de serveurs, ne vous donnant pas la possibilité de fixer arbitrairement les en-têtes HTTP (comme `Expires` et `Expires`), réclamez-la avec force ; ce sont des outils indispensables pour faire votre travail.

Les en-têtes HTTP sont envoyées par le serveur, avant le HTML, et sont seulement vues par le navigateur et les caches intermédiaires. Une réponse HTTP 1.1 typique se présenterait comme ceci :

```
HTTP/1.1 200 OK
Date: Fri, 30 Oct 1998 13:19:41 GMT
Server: Apache/1.3.3 (Unix)
Cache-Control: max-age=3600, must-revalidate
Expires: Fri, 30 Oct 1998 14:19:41 GMT
Last-Modified: Mon, 29 Jun 1998 02:28:12 GMT
ETag: "3e86-410-3596fbbc"
Content-Length: 1040
Content-Type: text/html
```

Le code HTML suit ces en-têtes séparé par une ligne vide. Cf. les sections concernant la mise en œuvre pour des informations sur le paramétrage des en-têtes HTTP.

Les en-têtes HTTP `Pragma` (et pourquoi elles ne fonctionnent pas)

Beaucoup de personnes croient qu'assigner une en-tête HTTP `Pragma: no-cache` à une représentation la rendra non cachable. Ce n'est pas toujours vrai : la spécification HTTP n'établit aucune directive pour les en-têtes de réponse `Pragma`, et traite plutôt des en-têtes de requête `Pragma` (les en-têtes envoyées au serveur par le navigateur). Bien que certains caches honorent cette en-tête, ce n'est pas le cas de la majorité, et elle n'aura aucun effet. Utilisez plutôt les en-têtes qui suivent.

Le contrôle de la fraîcheur avec l'en-tête HTTP `Expires`

L'en-tête HTTP `Expires` représente un moyen élémentaire pour contrôler les caches : elle indique à tous les caches pour combien de temps la représentation associée reste fraîche. Après échéance, les caches vérifieront toujours auprès du serveur original si le document a changé. Les en-têtes `Expires` sont reconnues par pratiquement tous les caches.

La plupart des serveurs Web permettent de régler les en-têtes de réponse `Expires` de plusieurs façons. Ils permettent communément de fixer une date d'expiration absolue, une date basée sur celle de la dernière représentaiton vue par le client (dernière *date d'accès*), ou une date basée sur celle de la dernière modification du document sur votre serveur (dernière *date de modification*.

Les en-têtes `Expires` conviennent tout particulièrement à la mise en cache des images statiques (comme les barres et boutons de navigation). Puisqu'elles ne changent pas beaucoup, vous pouvez leur donner des dates d'expiration extrêmement éloignées, en rendant votre site beaucoup plus réactif à vos utilisateurs. Les en-têtes sont aussi utile pour contrôler la mise en cache d'une page qui change régulièrement. Par exemple, si vous mettez quotidiennement à jour une page de nouvelles à six heures du matin, vous pouvez fixer l'expiration de la représentation à cette heure, de sorte que les caches sauront quand obtenir une copie fraîche, sans que les utilisateurs aient à cliquer sur « recharger ».

La **seule** valeur admise dans une en-tête `Expires` est une date HTTP, tout autre chose sera probablement interprété comme une date « dans le passé », la représentation n'étant donc pas cachable. Rappelez-vous aussi que l'heure d'une date HTTP est relative au temps GMT, non au temps local.

Par exemple :

```
Expires: Fri, 30 Oct 1998 14:19:41 GMT
```

Quoique l'en-tête `Expires` soit utile, elle a quelques limitations. D'abord, puisqu'il est question d'une date, les horloges du serveur Web et du cache doivent être synchronisées ; si leurs interprétations du temps sont différentes, on n'obtiendra pas le résultat escompté, et les caches pourraient, à tort, estimer frais un contenu périmé.

Un autre problème avec `Expires` est qu'il est facile d'oublier que l'on a fixé une date d'expiration particulière à un certain contenu. Si vous ne réactualisez pas une date `Expires` avant échéance, toutes les requêtes iront à votre serveur Web, en augmentant la charge et le temps d'attente.

> Il importe de s'assurer que l'horloge de votre serveur Web soit juste pour utiliser l'en-tête `Expires`. Un moyen d'y parvenir consiste à utiliser le protocole de date de réseau (NTP) ; parlez-en à votre administrateur de système local pour en savoir plus.

Les en-têtes HTTP `Cache-Contro`

HTTP 1.1 a introduit la nouvelle classe d'*en-têtes de réponse Cache-Control* afin de donner plus de contrôle aux éditeurs Web sur leur contenu et de corriger les limitations de l'en-tête `Expires`.

Les en-têtes de réponse `Cache-Control` utiles sont les suivantes :

- **max-age=**[secondes] — indique la quantité de temps maximale où la représentation sera considérée fraîche. Similaire à `Expires`, cette directive est relative à l'heure de la requête au lieu d'être absolue. La valeur [secondes] est le nombre de secondes, à compter de l'heure de la requête, pour lequel vous désirez que la

représentation soit fraîche.
- **s-maxage=**[secondes] — similaire à max-age, sauf qu'elle ne s'applique qu'aux caches partagés (par exemple, un mandataire).
- **public** — marque les réponses authentifiées comme cachables ; normalement, si une authentification HTTP est demandée, les réponses sont automatiquement non cachables.
- **no-cache** — force à chaque fois les caches à soumettre la requête au serveur original pour validation avant libération d'une copie cachée. C'est utile pour s'assurer du respect de l'authentification (en combinaison avec `public`), ou pour maintenir une fraîcheur rigide, sans sacrifier tous les bénéfices de la mise en cache.
- **no-store** — instruit les caches de ne pas garder de copie de la représentation dans toutes les conditions.
- **must-revalidate** — indique aux caches de devoir obéir à toute information de fraîcheur que vous leur donnez à propos d'une représentation. HTTP permet aux caches de servir des représentations périmées sous certaines conditions ; en indiquant cette en-tête, vous dites au cache que vous voulez un respect strict de vos règles.
- **proxy-revalidate** — similaire à `must-revalidate`, sauf qu'elle ne s'applique qu'aux caches de mandataires.

For example:

```
Cache-Control: max-age=3600, must-revalidate
```

Si vous pensez utiliser les en-têtes `Cache-Control`, vous devriez consulter l'excellente documentation dans HTTP 1.1 ; cf. Références et autres informations.

Les validateurs et la validation

Dans la section Comment fonctionnent les caches Web, nous disions que les serveurs et les caches utilisaient une validation pour communiquer le fait qu'une représentation avait changé. Cette validation permet d'éviter aux caches de devoir télécharger la représentation entière, lorsqu'ils en ont une copie locale mais dont ils ne sont pas sûrs de la fraîcheur.

Les validateurs sont très importants : si aucun n'est présent et si aucune information de fraîcheur (`Cache-Control` ou `Expires`) n'est disponible, les caches ne stockeront aucune représentation du tout.

Le validateur le plus courant est la date de dernière modification du document, communiquée par l'en-tête `Last-Modified`. Lorsque le cache stocke une représentation incluant une en-tête `Last-Modified`, il peut s'en servir pour interroger le serveur, avec une requête `If-Modified-Since`, afin de savoir si la représentation a changé depuis la dernière fois où elle a été vue.

HTTP 1.1 a introduit un nouveau type de validateur appelé `ETag`. Les `ETag` sont des identificateurs uniques générés par le serveur et changés à chaque fois que la représentation change. Puisque le serveur contrôle la génération de `Etag`, les caches ont l'assurance que, si l'`Etag` correspond à leurs requêtes `If-None-Match`, la représentation est réellement la même.

Presque tous les caches se servent de dates `Last-Modified` pour déterminer si la représentation est fraîche ; la validation `Etag` devient aussi prévalente.

La plupart des serveurs Web modernes généreront automatiquement à la fois des validateur `ETag` et `Last-Modified` pour du contenu statique (c.à.d. des fichiers) ; vous n'aurez rien à faire. Par contre, pour du contenu dynamique (comme les sites avec CGI, ASP et bases de données), ils n'en savent pas assez pour les générer ; cf. Écrire des scripts compatibles avec les caches.

Astuces pour construire un site compatible avec les caches

Hormis utiliser des informations de fraîcheur et une validation, il y a beaucoup de choses à faire pour rendre votre site plus convivial aux caches :

- **Utilisez des adresse URL de manière cohérente** — c'est la règle d'or de la mise en cache. Si vous servez le même contenu dans des pages différentes, à des utilisateurs différents, ou depuis des sites différents, il devrait avoir la même adresse URL. C'est le moyen le plus facile et le plus efficace pour rendre votre site convivial aux caches. Par exemple, si vous utilisez une première fois « /index.html » dans votre code HTML comme appel, utilisez-le toujours de cette façon ;
- **Utilisez une bibliothèque commune d'images** et d'autres éléments, et appelez-les depuis des endroits différents ;
- **Faites que les caches stockent les images et les pages ne changeant pas souvent** en utilisant une en-tête `Cache-Control: max-age` avec une grande valeur ;

- **Faites que les caches reconnaissent les pages mises à jour régulièrement** en indiquant un max-age ou une date d'expiration appropriés ;
- **Si une ressource (un fichier téléchargeable spécialement) change, changez son nom**. De cette manière, vous pouvez la faire expirer loin dans le future, tout en garantissant le service de la version correcte ; seule la page qui mène à elle a besoin d'une date d'expiration proche ;
- **Ne changez pas les fichiers sans nécessité**. Si vous le faites, tout aura une date `Last-Modified` faussement récente. Par exemple, lors d'une mise à jour de votre site, ne recopiez pas le site entier ; déplacez juste les fichiers que vous aurez changé ;
- **Utilisez des cookies seulement si nécessaire** — les cookies sont difficiles à cacher et ne sont pas nécessaires dans la plupart des situations. Si vous devez utiliser un cookie, limitez-en l'usage aux pages dynamiques ;
- **Minimisez l'utilisation de SSL** — parce que les pages chiffrées ne sont pas stockées par les caches partagés, ne les utilisez que quand il le faut, et utilisez avec parcimonie les images dans les pages SSL ;
- **Utilisez le Cacheability Engine** — il peut vous aider à mettre en pratique plusieurs des concepts de ce tutoriel.

Écrire des scripts compatibles avec les caches

Par défaut, la plupart des scripts ne retourneront pas de validateur (par exemple, une en-tête HTTP `Last-Modified`, ou `Etag`) ni d'informations de fraîcheur (`Cache-Control` ou `Expires`). Tandis que quelques scripts sont vraiment dynamiques (c'est-à-dire qu'ils retournent une réponse différente à chaque requête), beaucoup (comme les moteurs de recherche et les sites menés par base de données) peuvent bénéficier à être conviviaux aux caches.

Généralement parlant, si un script produit une sortie reproductible pour la même requête à une date ultérieure (qu'il s'agisse de minutes ou de jours plus tard), elle devrait être cachable. Si le contenu du script change seulement en fonction du contenu de l'adresse URL, il est cachable ; si la sortie dépend d'un cookie, d'informations d'authentification ou d'autres critères externes, elle ne l'est probablement pas.

- La meilleure façon de rendre un script convivial aux caches (ainsi que plus performant) est de vider son contenu dans un fichier simple à chaque fois qu'il change. Le serveur Web peut alors le traiter comme n'importe quelle autre page Web, en générant et utilisant des validateurs, ce qui facilite votre vie. Rappelez-vous d'écrire seulement les fichiers qui auront changés, pour préserver les dates `Last-Modified`.
- Une autre façon de rendre un script cachable dans une certaine mesure est de fixer une en-tête d'âge relatif aussi loin dans le futur que pratique. Bien qu'on puisse y parvenir avec `Expires`, il est probablement plus facile de le faire avec `Cache-Control: max-age`, ce qui gardera la représentation fraîche pendant un certain laps de temps après la requête.
- Si vous ne pouvez pas le faire, il faudra faire générer un validateur au script, et répondre ensuite aux requêtes `If-Modified-Since` et/ou `If-None-Match`. On y parvient en analysant les en-têtes HTTP et en répondant alors avec un code 304 Non Modifié quand c'est approprié. Ce n'est malheureusement pas une tâche facile.

Quelques autres astuces :

- **N'utilisez pas POST** sauf si nécessaire. Les réponses de la méthode POST ne sont pas conservées par la plupart des caches ; si vous passez des informations par le chemin ou la requête (via GET), les caches peuvent stocker ces informations pour la suite ;
- **N'incorporez pas d'informations spécifiques de l'utilisateur dans l'adresse URL** à moins que le contenu généré soit complètement unique à cet utilisateur ;
- **Ne comptez pas sur le fait que toutes les requêtes d'un utilisateur proviennent d'un seul hôte** parce que les caches travaillent souvent de concert ;
- **Générez des en-têtes de réponse `Content-Length`**. C'est facile à faire et permet d'utiliser la réponse de votre script dans une *connexion persistente*. Cela permet aux clients de demander plusieurs représentations en une seule connexion TCP/IP, au lieu d'établir une connexion à chaque requête. Votre site semblera plus rapide.

Cf. les Notes de mise en œuvre pour des précisions.

Foire aux questions

Quelles choses les plus importantes faut-il rendre cachables ?

Une bonne stratégie consiste à identifier les représentations les plus volumineuses et les plus populaires (en particulier les images) et de commencer d'abord par elles.

Comment rendre mes pages aussi vives que possibles avec les caches ?

La représentation la plus cachable est celle dont la durée de fraîcheur est longue. La validation aide bien à réduire le temps pris pour voir une représentation, mais le cache doit quand même contacter le serveur original pour connaître sa fraîcheur. Si le cache la connaît déjà, la représentation sera servie directement.

Je comprend que la mise en cache est bonne, mais j'ai besoin de tenir des

statistiques sur le nombre de personnes qui visitent ma page !

Si vous devez connaître le nombre d'accès à une page, sélectionnez UN SEUL petit élément de celle-ci (ou la page même), et rendez-la cachable en lui donnant les en-têtes appropriées. Par exemple, vous pourriez appeler une image 1 × 1 transparente non cachable dans chaque page. L'en-tête Referer contiendra des informations à propos de la page qui l'aura appelée.

Sachez que même cette méthode ne donnera pas de statistiques vraiment précises concernant vos utilisateurs, et elle nuit à l'Internet et à vos utilisateurs, car elle génère du trafic superflu et force les personnes à attendre le téléchargement de l'élément non caché. Pour plus de renseignements à ce sujet, cf. « On Interpreting Access Statistics » dans les références.

Comment puis-je voir les en-têtes HTTP d'une représentation ?

Beaucoup de navigateurs Web vous permettent de voir la valeur des en-têtes `Expires` et `Last-Modified` dans des « Informations sur la page » ou une interface similaire. Si c'est le cas, vous obtiendrez un menu de la page et de toutes les représentations (comme les images) qui lui sont associées, en même temps que les détails les concernant.

Pour voir les en-têtes complètes d'une représentation, vous pouvez vous connecter manuellement au serveur Web en utilisant un client Telnet.

Pour cela, il vous faudra peut-être saisir le port (par défaut, le port 80) dans un champs à part, ou vous connecter au serveur par www.myhost.com:80 ou www.myhost.com 80 (notez l'espace). Consultez la documentation de votre client Telnet.

Une fois la connexion au site établie, tapez une requête pour la représentation. Par exemple, si vous voulez voir les en-têtes de http://www.myhost.com/foo.html, connectez-vous à to www.myhost.com, port 80, et tapez :

```
GET /foo.html HTTP/1.1 [retour]
Host: www.myhost.com [retour][retour]
```

Appuyez la touche « Retour chariot » pour chaque [retour] ; assurez-vous de l'appuyer deux fois à la fin. Cela affichera les en-têtes puis la représentation entière. Pour les en-têtes seulement, remplacez GET par HEAD.

Mes pages sont protégées par un mot de passe. Comment les caches de mandataires s'en accomodent-ils ?

Par défaut, les pages protégées par authentification HTTP sont considérées comme privées ; les caches partagés ne les conserveront pas. Toutefois, vous pouvez rendre publiques des pages authentifiées avec une en-tête `Cache-Control:÷public` ; les caches compatibles avec HTTP 1.1 seront alors autorisés à les mettre en cache.

Si vous souhaitez que ces pages soient cachables, mais toujours authentifiée pour chaque utilisateur, combinez les en-têtes `Cache-Control: public` et `no-cache`. Cela ordonne au cache de soumettre les informations d'authentification du nouveau client au serveur original avant de libérer la représentation du cache. Voici à quoi ça ressemble :

```
Cache-Control: public, no-cache
```

Que ce soit le cas ou non, il vaut mieux minimiser l'utilisation de l'authentification ; par exemple, si vos images ne sont pas sensibles, placez-les dans un répertoire séparé et configurez votre serveur afin qu'il ne force pas l'authentification pour celui-ci. Les images seront ainsi naturellement cachables.

Dois-je m'inquiéter de la sécurité si des personnes accèdent à mon site au travers d'un cache ?

Les pages SSL ne sont pas cachées (ou déchiffrées) par les caches de mandataires, et vous ne devez donc pas vous en soucier. Par contre, puisque les caches stockent les requêtes non-SSL et les adresses URL récupérées par leur biais, vous devriez en tenir compte pour les sites non sécurisés ; un administrateur sans scrupule peut théoriquement réunir des informations concernant leurs utilisateurs, en particulier dans l'adresse URL.

En fait, n'importe quel administrateur sur le réseau entre votre serveur et vos clients pourrait réunir ce type d'informations. Un problème particulier se pose lorsque les scripts CGI inscrivent les identifiants et les mots de passe dans l'adresse URL même ; il devient très facile à d'autres de trouver et d'utiliser cette identification.

Si vous connaissez les problèmes entourant la sécurité sur le Web en général, vous ne devriez pas avoir de surprises de la part des caches de mandataires.

Je recherche une solution de publication Web intégrée. Quelles sont celles compatibles avec les caches ?

Ça dépend. Généralement parlant, plus la solution est complexe, plus elle est difficile à cacher. Les moins bonnes

sont celles qui génèrent dynamiquement tout le contenu et ne fournissent pas de validateurs ; elles sont peut-être pas du tout cachable. Renseignez-vous auprès de l'équipe technique du vendeur, et voyez les « Notes de mise en œuvre » ci-dessous.

Mes images expirent dans un mois à partir de maintenant, mais je dois les changer dans le cache tout de suite !

L'en-tête `Expires` peut être contournées ; la copie en cache servira tant que le cache (du navigateur, ou bien du mandataire) ne manque pas de place et ne supprime les représentations, .

La solution la plus efficace consiste à changer tous les liens vers elles ; des représentations complètement nouvelles seront ainsi chargées fraiches du serveur original. Rappelez-vous que la page appelant la représentation sera aussi mise en cache. À cause de ça, il vaut mieux rendre les images statiques et les représentations similaires très cachables, en gardant un lien étroit sur les pages HTML qui les appellent

Si vous souhaitez recharger une représentation d'un cache particulier, vous pouvez soit forcer le rechargement (dans Netscape, maintenir la touche majuscule appuyée tout en pressant « Rafraîchir ») le fera en envoyant une en-tête de requête `Pragma: no-cache`) au cours de l'utilisation du cache. Ou bien vous pouvez demander à l'administrateur du cache, par l'intermédiaire d'une interface, de supprimer la représentation.

J'exploite un service d'hébergement Web. Comment faire pour que mes utilisateurs publient des pages conviviales aux caches ?

Si vous utilisez Apache, laissez-les employer des fichiers .htacces en leur fournissant une documentation appropriée.

Sinon vous pouvez établir des zones prédéterminées pour divers attributs de mise en cache dans chaque serveur virtuel. Par exemple, vous pourriez définir un répertoire « /cache-1m » à mettre en cache pour un mois après accès, et une zone « /no-cache » qui sera servie avec des en-têtes instruisant les caches de ne pas stocker les représentations qui en proviennent.

Quoi que vous fassiez, il vaut mieux travailler d'abord avec vos clients les plus importants pour la mise en cache. L'essentiel des économies réalisées (en bande passante et en charge des serveurs) viendra des sites à gros volumes.

J'ai marqué mes pages comme cachables, mais mon navigateur continue à les demander à chaque requête. Comment forcer le cache à en garder des représentations ?

Les caches sont tenus de conserver une représentation et de la réutiliser ; ils sont seulement tenus de **ne pas** les garder ou les utiliser sous certaines conditions. Tous les caches prennent des décision concernant les représentations à garder en fonction de leur dimension, leur type (par exemple, image ou html), ou de l'espace restant pour conserver des copies locales. Vos représentations seront peut-être jugées d'intérêt moindre à garder comparées à des représentations plus populaires ou plus volumineuses.

Certains caches offrent bien à leurs administrateurs la possibilité de privilégier les types de représentations à garder, d'autres d'« épingler » les représentations dans le cache, pour qu'elles soient toujours disponibles.

Notes de mise en œuvre — Les serveurs Web

Généralement parlant, il vaut mieux utiliser la dernière version du serveur Web, quel qu'il soit, que vous avez choisi de déployer. Non seulement la nouvelle version présentera vraisemblablement plus de fonctionnalités conviviales pour les caches mais habituellement elle comportera aussi des améliorations importantes de la sécurité et des performances.

Le serveur HTTP Apache

Apache se sert de modules optionnels pour inclure les en-têtes, dont `Expires` et `Cache-Control`. Les deux modules sont disponibles à partir de la distribution version 1.2.

Les modules doivent être construits dans Apache ; bien qu'inclus dans la distribution, ils ne sont pas activés par défaut. Pour voir si les modules sont opérationnels dans votre serveur, cherchez le programme httpd et exécutez `httpd -l`, ce qui devrait afficher la liste des modules disponibles. Les modules à rechercher sont mod_expires et mod_headers.

- S'ils ne sont pas disponibles et que vous ayez un accès de niveau administrateur, vous pouvez recompiler Apache pour les inclure. On y parvient soit en décommentant les lignes appropriées dans le fichier de configuration, soit en utilisant les arguments de configuration `-enable-module=expires` et `-enable-module=headers` (à partir de la version 1.3). Cf. le fichier INSTALL compris dans la distribution Apache.

Apache et les modules appropriés une fois prêt, vous pouvez utiliser mod_expires pour définir quand doivent expirer les représentations, soit dans les fichiers .htaccess, soit dans le fichier access.conf du serveur. Vous pouvez définir l'expiration soit d'après la date d'accès, soit d'après la date de modification, et l'appliquer à un type de fichier ou par défaut. Cf. la documentation du module pour des précisions, et voyez votre gourou Apache local si vous avez des problèmes.

Pour appliquer les en-têtes `Cache-Control`, vous vous servirez du module mod_headers qui permet de définir les en-têtes HTTP arbitraires d'une ressource. Cf. la documentation de mod_headers.

Voici un fichier .htaccess d'exemple pour illustrer l'utilisation de quelques en-têtes :

- Les fichiers .htaccess permettent aux éditeurs Web d'utiliser des commandes qui ne se trouvent normalement que dans les fichiers de configuration. Ils agissent sur le contenu du répertoire où ils se trouvent et ses sous-répertoires. Parlez-en à l'administrateur de votre serveur pour savoir s'ils sont activés.

```
### activate mod_expires
ExpiresActive On
### Expire les .gif un mois après leur accès
ExpiresByType image/gif A2592000
### Expire tout le reste un jour après sa dernière modification
### (utilisation de la syntaxe alternative)
ExpiresDefault "modification plus 1 day"
### Applique une en-tête Cache-Control à index.html
<Files index.html>
Header append Cache-Control "public, must-revalidate"
</Files>
```

- Remarquez que mod_expires calcule et insère automatiquement une en-tête `Cache-Control:max-age` si nécessaire.

La configuration d'Apache 2.0 est très semblable à celle de la version 1.3 ; cf. la documentation de mod_expires et de mod_headers de la version 2.0 pour des précisions.

Microsoft IIS

Le serveur Internet Information Server (IIS) de Microsoft permet très facilement de régler les en-têtes de manière quelque peu flexible. Remarquez que ce n'est possible qu'avec la version 4 du serveur, qui fonctionne seulement sur NT Server.

Pour définir les en-têtes d'une zone du site, sélectionnez-la dans l'interface « Outils d'administration » et faites apparaître ses propriétés. Après avoir sélectionné l'onglet « En-têtes HTTP », vous devriez voir deux zones intéressantes : « Activer l'expiration de contenu » et « En-têtes HTTP personnalisées ». La première ne nécessite aucune explication, et la seconde peut servir à appliquer des en-têtes `Cache-Control`.

Voir la section ASP suivante pour des informations concernant le réglage des en-têtes dans Active Server Pages. Il est également possible de régler les en-têtes à partir des modules ISAPI (cf. MSDN pour des précisions).

Netscape/iPlanet Enterprise Server

En ce qui concerne la version 3.6, le serveur Enterprise Server n'offre aucun moyen évident de fixer les en-têtes `Expires`. Par contre, il gère les caractéristiques HTTP 1.1 depuis la version 3.0. Les caches HTTP 1.1 (mandataire et navigateur) pourront donc tirer partie des réglages `Cache-Control` que vous ferez.

Pour utiliser les en-têtes `Cache-Control`, choisissez « Gestion de contenu | Directives de contrôle des caches » dans le serveur d'administration. Puis, en vous servant du « Préleveur de ressource » (N.d.T. Resource Picker), choisissez le répertoire où vous voulez régler les en-têtes. Après le réglage, cliquez « OK ». Pour plus de renseignements, cf. le manuel NES.

Notes de mise en œuvre — Les scripts côté-serveur

> Une chose à garder à l'esprit c'est qu'il est peut-être plus facile de régler les en-têtes HTTP auprès de votre serveur Web qu'avec le langage de script. Essayez les deux.

Puisqu'avec les scripts côté-serveur l'accent est mis sur le contenu dynamique, cela ne donne pas de pages très cachables, même si le contenu pourrait l'être. Si votre contenu change souvent, mais pas à chaque accès de la page, pensez à placer une en-tête `Cache-Control: max-age` ; la plupart des utilisateurs reviennent sur les pages au cours d'une période de temps relativement courte. Par exemple, lorsque les utilisateurs appuient le bouton « retour », si aucun validateur ni information de fraîcheur sont disponibles, ils devront attendre le rechargement de la page depuis le serveur pour la voir.

CGI

Les scripts CGI sont l'un des moyens les plus populaires de générer du contenu. Vous pouvez facilement adjoindre des en-têtes de réponse HTTP en les ajoutant avant d'envoyer le corps. La plupart des mises en œuvre CGI vous imposent déjà de le faire pour l'en-tête `Content-Type`. Par exemple, en Perl :

```
#!/usr/bin/perl
print "Content-type: text/html\n";
print "Expires: Thu, 29 Oct 1998 17:04:19 GMT\n";
print "\n";
### le corps du contenu suit ...
```

Puisque ce n'est que du texte, vous pouvez facilement générer des en-têtes `Expires` et d'autres relatives à une date avec les fonctions intégrées. C'est encore plus facile si vous utilisez `Cache-Control: max-age` :

```
print "Cache-Control: max-age=600\n";
```

Cela rendra le script cachable pour les dix minutes suivant la requête, de sorte que, si l'utilisateur presse le bouton « retour », la requête ne sera pas soumise à nouveau.

La spécification CGI fait aussi que les en-têtes de requêtes envoyées par le client sont disponibles dans l'environnement du script ; la chaîne « HTTP_ » s'ajoute au début de chaque nom d'en-tête. Donc, si un client fait une requête `If-Modified-Since`, elle apparaîtra comme ceci :

```
HTTP_IF_MODIFIED_SINCE = Fri, 30 Oct 1998 14:19:41 GMT
```

Cf. également la bibliothèque cgi_buffer, qui gère automatiquement la génération et la validation des en-têtes `ETag`, la génération de `Content-Length` et la compression `Content-Encoding: gzip` des scripts CGI pour Perl et Python en une ligne include. La version Python peut aussi servir pour envelopper des scripts CGI arbitraires.

Les inclusions côté-serveur (SSI)

Le script SSI (souvent utilisé avec l'extension .shtml) est l'un des premiers moyens ayant permis aux éditeurs Web d'amener du contenu dynamique dans les pages. On avait une forme limitée de script dans HTML en utilisant des balises spéciales dans les pages.

La plupart des mises en œuvre SSI ne fixent pas de validateurs et, de ce fait, ne sont pas cachables. Par contre, les mises en œuvre Apache permettent aux utilisateurs d'indiquer quels fichiers SSI peuvent l'être, en réglant les permissions d'exécution de groupe sur les fichiers appropriés, combiné à la directive complète `XbitHack`. Pour plus de renseignements, cf. la documentation mod_include.

PHP

PHP est un langage de script côté-serveur qui, si construit dans le serveur, peut servir à incorporer des scripts au sein du code HTML de la page, un peu comme SSI, mais avec beaucoup plus d'options. PHP peut être utilisé comme un script CGI sur n'importe quel serveur Web (Unix ou Windows), ou comme un module Apache.

Par défaut, les représentations traitées par PHP n'ont pas de validateurs et sont donc non cachables. Toutefois, les développeurs peuvent fixer des en-têtes HTTP au moyen de la fonction `header()`.

Par exemple, ce code créera une en-tête `Cache-Control` ainsi qu'une en-tête `Expires` valide pendant trois jours :

```
<?php
 Header("Cache-Control: must-revalidate");

 $offset = 60 * 60 * 24 * 3;
 $ExpStr = "Expires: " . gmdate("D, d M Y H:i:s", time() + $offset) . " GMT";
 Header($ExpStr);
?>
```

Rappelez-vous que la function `header()` DOIT apparaître avant toute autre sortie.

Comme vous le voyez, il faudra créer manuellement la date HTTP d'une en-tête `Expires` ; PHP n'offre aucune fonction qui le fasse pour vous (quoique les versions récentes l'ont rendue plus aisée, cf. la documentation de date de PHP). Bien sûr, il est facile de fixer une en-tête `Cache-Control: max-age`, qui est tout aussi bonne pour la plupart des situations.

Pour plus de renseignements, cf. l'entrée dans le manuel de header.

Cf. également la bibliothèque cgi_buffer, qui gère automatiquement la génération et validation des en-têtes `ETag`, la génération de `Content-Length` et la compression `Content-Encoding: gzip` pour les scripts PHP en une seule ligne include.

Cold Fusion

Cold Fusion, de Adobe est un moteur de script côté-serveur commercial, fonctionnant avec plusieurs serveurs Web sur Windows, Linux et plusieurs variantes Unix.

Cold Fusion rend le réglage d'en-têtes HTTP arbitraires relativement aisé, avec la balise `CFHEADER`. Malheureusement, leur exemple de réglage d'une en-tête `Expires`, comme ci-dessous, est un peu trompeur.

```
<CFHEADER NAME="Expires" VALUE="#Now()#">
```

Ça ne fonctionne pas comme on pourrait le croire, parce que le temps (dans ce cas, lorsque la requête est faite) n'est pas converti en une date HTTP valide ; à la place, elle s'affiche juste comme une représentation de l'objet Date/Time de Cold Fusion. La plupart des clients soit ignoreront cette valeur, soit la convertiront en une valeur par défaut, comme « January 1, 1970 ».

Toutefois, Cold Fusion offre une fonction de formatage de date qui fera le travail : `GetHttpTimeString`. En combinaison avec la fonction `DateAdd`, il est facile de fixer des dates `Expires` ; nous fixons ici une en-tête déclarant que les représentations de la page expireront dans un mois :

```
<cfheader name="Expires" value="#GetHttpTimeString(DateAdd('m', 1, Now()))#">
```

Vous pouvez aussi vous servir de la balise CFHEADER pour fixer les en-têtes `Cache-Control: max-age` et d'autres.

Rappelez-vous que les en-têtes des serveurs Web transitent (N.d.T. pass through) dans certains déploiements de Cold Fusion (tel que CGI) ; vérifiez le vôtre pour déterminer si vous pouvez vous en servir avantageusement, en réglant les en-têtes sur le serveur plutôt que dans Cold Fusion.

ASP

Les scripts ASP, construits dans IIS et disponibles aussi pour d'autres serveurs Web, vous permettent de régler les en-têtes HTTP. Par exemple, pour fixer une date d'expiration, vous pouvez utiliser les propriétés de l'objet `Response` :

```
<% Response.Expires=1440 %>
```

En indiquant le nombre de minutes à compte de la requête pour faire expirer la représentation. De même, on peut fixer une date d'expiration absolue de cette façon (assurez-vous de formater correctement la date HTTP) :

```
<% Response.ExpiresAbsolute=#May 31,1996 13:30:15 GMT# %>
```

On peut ajouter des en-têtes `Cache-Control` comme ceci :

```
<% Response.CacheControl="public" %>
```

> Lors du réglage d'en-têtes HTTP depuis un script ASP, assurez-vous de placer les appels de la méthode `Response` avant toute génération HTML, ou bien d'utiliser `Response.Buffer` pour mettre en tampon la sortie. Notez aussi que certaines versions du serveur IIS fixe une en-tête `Cache-Control: private` sur les scripts ASP par défaut, qui doivent être déclarés `public` pour être cachables par les caches partagés.

Dans ASP.NET, `Response.Expires` est déconseillé : la bonne façon de régler les en-têtes relatives aux caches est avec `Response.Cache` :

```
Response.Cache.SetExpires ( DateTime.Now.AddMinutes ( 60 ) ) ;
Response.Cache.SetCacheability ( HttpCacheability.Public ) ;
```

Cf. la documentation MSDN pour plus de renseignements.

Références et autres informations

Spécification HTTP 1.1

La spécification HTTP 1.1 décrit beaucoup d'extension pour rendre les pages cachables et c'est le guide qui fait autorité pour la mise en œuvre du protocole. Cf. les sections 13, 14.9, 14.21 et 14.25.

Web-Caching.com

Une excellente introduction aux concepts de la mise en cache, avec des liens vers d'autres ressources en ligne.

On Interpreting Access Statistics

Une déblatération informative de Jeff Goldberg selon laquelle pourquoi vous ne devriez pas compter sur les statistiques d'accès et les compteurs d'appels de fichiers.

Cacheability Engine

En examinant les pages Web afin de déterminer comment elles interagissent avec les caches Web, le moteur est un bon outil de débogage et un auxilliaire de ce tutoriel.

Bibliothèque cgi_buffer

Une seule ligne include en Perl CGI, Python CGI et scripts PHP, qui gère automatiquement la génération et la validation des en-têtes `ETag`, la génération `Content-Length` et la compression `Content-Encoding: gzip`, et cela correctement. La version Python peut également servir d'enveloppe autour de scripts CGI arbitraires.

À propos de ce document

Si vous hébergez ce document en miroir, veuillez envoyer un courrier à l'adresse précedente afin d'être informé des mises à jour.

La dernière révision de ce document est toujours disponible à http://www.mnot.net/cache_docs/

Traductions disponibles : tchèque.

Version 1.7 — May 9, 2006