SEO et fichier .htaccess : les erreurs à éviter

Par Aymeric Bouillat

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

Le fichier .htaccess est une composante souvent essentielle des serveurs Apache. Très souvent utilisé pour mettre par exemple en place des redirections, elle est également source d'erreurs fréquentes ou de mauvaises utilisations qui peuvent avoir un impact non négligeable sur les performances de votre serveur. Cet article liste un certain nombre d'erreurs le plus souvent rencontrées et explique la meilleure façon de les corriger.

Que ce soit pour effectuer de la réécriture afin d'obtenir des URL SEO-friendly ou des redirections 301 en cas de modifications d'URL, on parle souvent de fichiers .htaccess. Ces fichiers textes placés à la racine de répertoires permettent d'inclure ce type de directives pour manipuler les URL avec aisance. Ils sont spécifiques aux serveurs Apache qui restent les serveurs Web les plus courants, bien que les serveurs Nginx gagnent du terrain ces derniers temps.

Il existe cependant des erreurs assez courantes de configuration des fichiers .htaccess. Nous allons détailler certaines d'entre elles, qui vous permettront de ne pas réduire les performances du serveur, mais aussi d'éviter la duplication de contenu ou des redirections ratées avec des règles mal formatées qui pourraient nuire à votre référencement et provoquer des effets négatifs lors du crawl de Googlebot.

Fonctionnement

Les fichiers .htaccess permettent de modifier la configuration du serveur Apache au niveau du répertoire. Bien qu'ils aient un côté pratique, car leur implémentation ne nécessite pas de redémarrage du serveur pour une bonne prise en compte des règles, l'Apache Software Fundation déconseille leur utilisation dans la mesure du possible, pour des questions de performance que nous allons détailler dans cet article.

Il est préférable de placer les directives dans les fichiers de configuration Apache côté serveur, plutôt que dans les .htaccess qui sont gourmands en ressources. On parle donc de fichiers .htaccess par abus de langage, mais l'utilisation du terme « fichier de configuration Apache » est plus adaptée.

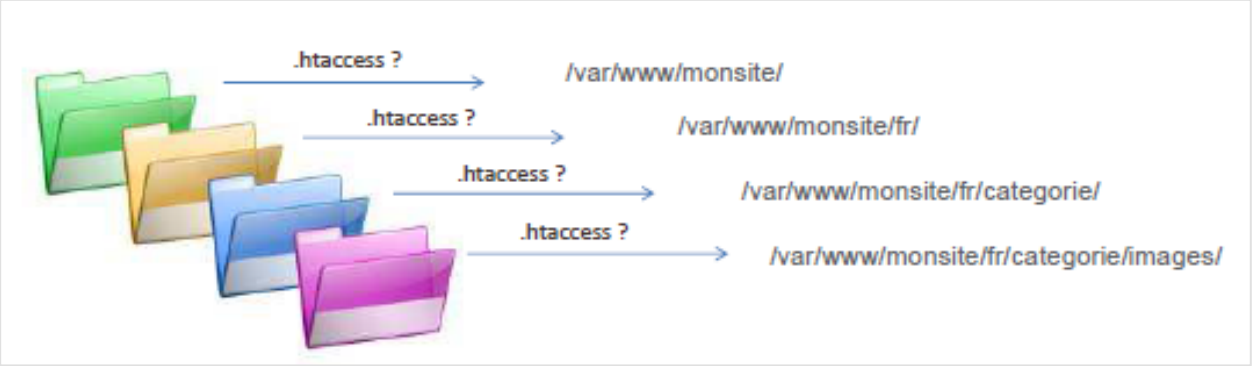

Fig.1. Vérification de la présence d'un fichier .htaccess dans l'arborescence

Prenons comme exemple le fonctionnement d'Apache avec une requête sur l'URL suivante : http://exemple.com/fr/categorie/images/fichier.jpg

Avant d'appeler le fichier physiquement lors de la requête, Apache va vérifier la présence d'un fichier .htaccess dans chaque répertoire, en partant du niveau supérieur (http://exemple.com/), vers le niveau le plus profond (/images/), tout en vérifiant sa présence dans les répertoires intermédiaires (/categorie/ et /fr/ dans notre exemple, voir figure 1).

Chaque fois qu'un fichier .htaccess est trouvé, les directives sont rassemblées : les plus profondes dans l'arborescence écrasent celles des niveaux supérieurs. Cette opération est répétée pour chaque requête, ce qui peut s'avérer gourmand en terme de ressources, d'autant plus quand vous avez des fichiers .htaccess composés de plusieurs centaines de lignes à interpréter pour chaque requête.

Dans l'exemple ci-dessus, pour chaque requête, Apache va effectuer des accès disques pour tester la présence de fichiers .htaccess dans les 4 répertoires.

L'équation peut vite faire exploser le nombre d'accès disque :

(Niveau de profondeur répertoires x Nombre de requêtes pour l'affichage d'une page) x Nombre d'internautes appelant le fichier = ***dégradation des performances****.*

L'ensemble des directives contenues dans les fichiers .htaccess doivent être placées directement dans la configuration du serveur quand cela est possible (hôte virtuelle ou fichier de configuration apache.conf/httpd.conf). Cela évitera également une recompilation des RewriteRule contenant des expressions rationnelles à chaque requête : elles seront compilées au chargement du serveur, puis mises en cache.

Le seul fait de placer ces directives directement dans la configuration d'Apache ne sera pas suffisant pour améliorer les performances de votre site Web. Il faudra désactiver la fonction qui teste la présence de fichier .htaccess. Cela se passe au niveau de la directive AllowOverride :

AllowOverride None
(au lieu de AllowOverride All)

	AllowOverride All (avec .htaccess)	AllowOverride None (sans .htaccess)
Time taken for tests	30.004 seconds	30.001 seconds
Complete requests	11038	21242
Requests per second	367.89 [#/sec] (mean)	708.04 [#/sec] (mean)
Time per request	13.591 [ms] (mean)	7.062 [ms] (mean)
Time per request	2.718 [ms] (mean, across all concurrent requests)	1.412 [ms] (mean, across all concurrent requests)

Fig.2. Test avec ou sans fichier .htaccess

Vous pourrez ensuite appliquer vos règles à des répertoires spécifiques comme avec les .htaccess via des sections <Directory> dans la configuration d'Apache pour cibler des fichiers spécifiques :

```
<Directory /var/www
/monsite/fr/categorie/images/>
#modification de la durée de mise en
cache des images
Header set Cache-Control "max-
age=2592000"
</Directory>
```

Plusieurs tests ont montré qu'Apache était parfois capable d'encaisser le double de requêtes/seconde avec des meilleurs temps de réponse, quand cette directive AllowOverride était désactivée. La figure 2 montre un exemple de résultats d'un test effectué avec l'outil Apache Benchmark (#ab).

En cas de pic de trafic, cette optimisation ne sera pas un luxe. A moins que vous effectuiez des modifications extrêmement régulières sur la configuration des règles Apache (auquel cas le recours aux .htaccess aura un côté pratique), il est préférable de faire implémenter directives et règles de réécriture au niveau du serveur, directement dans la configuration Apache, par votre DSI ou votre hébergeur. Au moment où les temps de chargement sont un élément clé en termes d'expérience utilisateur mais aussi de SEO, les fichiers .htaccess sont donc à utiliser avec modération.

Principales directives pour les redirections

Redirections, mod_alias ou mod_rewrite ?

Le module mod_alias supporte plusieurs directives dont nous verrons le fonctionnement avec différents exemples. La directive *Redirect* permet d'effectuer différents types de redirection. Elle s'utilise de la façon suivante :

```
Redirect typederedirection /ancienneurl
/nouvelleurl
```

Le paramètre type de redirection peut prendre plusieurs valeurs : *permanent* (301), *temp* (302), *seeother* (303) et enfin *gone* (410) pour laquelle le dernier argument ne sera donc pas nécessaire. **Attention : si le type de redirection n'est pas spécifié, c'est une redirection 302 qui sera effectuée par défaut.**

Il est donc préférable, quand on souhaite mettre en place des redirections page à page de type 301,

d'utiliser *RedirectPermanent* qui est spécifique aux redirections 301 :
RedirectPermanent /ancienneurl /nouvelleurl

Si vous avez des redirections plus génériques (et non pas page à page) en fonction d'un format d'URL bien défini, la directive *RedirectMatch* sera votre alliée. Exemple dans le cas d'une modification de nom de répertoire :
RedirectMatch /forum/(.*) /nouveauforum/$1 [R=301,L]
Cette règle aura pour effet de rediriger toutes les URL contenant /forum/quelquechose vers /nouveauforum/quelquechose, le signe point « . » représentant n'importe quel caractère, et l'étoile « * » indique que le caractère précédent peut être répété de 0 à plusieurs fois (soit plusieurs fois n'importe quel caractère). $1 ici reprendra les caractères compris dans le premier ensemble défini avec les parenthèses.

RewriteRule pour les redirections complexes

Bien que les directives fournies par le module mod_alias couvrent en général les principaux besoins en terme de redirection, elles ne vous permettent pas de manipuler les chaînes de paramètres des URL (partie située après le ? dans une url et constituée de couples paramètre/valeur sous la forme param1=valeur&param2=autrevaleur).

Si une chaîne de paramètres est présente dans l'URL à rediriger avec l'une des règles du module mod_alias, elle sera conservée dans l'URL de destination. De la même manière, si vous devez effectuer des redirections en fonction d'éléments spécifique (IP, nom d'hôte, port, etc.), vous ne pourrez pas le faire avec Redirect ou RedirectMatch.

La directive à utiliser dans ce type de cas est *RewriteRule* du module mod_rewrite. Vous allez pouvoir manipuler les chaînes de paramètres avec cette dernière, et effectuer vos redirections en fonction de nombreux paramètres d'environnements.

Elle est également capable de gérer de simples redirections 301, mais elle n'est pas recommandée dans ce cas, les directives *Redirect, RedirectPermanent* et *RedirectMatch* de mod_alias étant plus adaptées aux tâches simples de redirections.

Ecriture des règles et expressions rationnelles

Les exemples de règles de réécriture ou de redirections que l'on peut trouver sur le web sont parfois incomplets, des erreurs sont fréquentes avec les directives *RedirectMatch* ou *RewriteRule*, qui prennent en charge les expressions rationnelles compatibles PCRE notamment (module mod_alias et module de réécriture mod_rewrite d'Apache).

Début et fin d'une chaîne de caractères

Pour une simple règle de redirection 301 d'un répertoire à un autre avec une expression régulière, on pourrait avoir ce type de règle comme vu précédemment:
RedirectMatch /forum/(.*) /nouveaurepertoire/$1 [R=301,L]

Hors cette règle est fausse. Nous parlons bien d'URL contenant /forum/

mais pas d'URL commençant par /forum/ : une URL du type /fr/images/forum/header.jpg se retrouverait redirigée vers /nouveaurepertoire/header.jpg avec la règle précédente, ce qui provoquera une erreur 404.

Dès qu'on utilise une directive Apache prenant en charge les expressions rationnelles permettant de définir des motifs d'URL, il est indispensable d'indiquer le début de la chaîne avec le signe ^, et la fin de la chaîne de caractères avec le signe $. La règle précédente correctement écrite sera donc :

```
RedirectMatch ^/forum/(.*)$
/nouveaurepertoire/$1 [R=301,L]
```

Ainsi, seuls les URL commençant par /forum/ seront redirigées. Il est fréquent de voir également des règles avec expressions non fermées : RewriteRule ^forum/ /nouveaurepertoire/ [R=301,L]. Dans ce cas, les URL contenant forum/abc, ou forum/def seront prises en charge par la règle, car aucun caractère ne définit la fin de l'expresssion régulière, on cible par erreur toutes les URL qui commencent par /forum/.

Echappement des symboles

Une autre erreur courante est d'oublier d'échapper le signe point. Comme nous l'avons vu précédemment, il représente n'importe quel caractère. Avec la règle suivante :

```
RewriteRule ^anciennepage.html$
/nouvellepage.html [R=301,L]
```

appeler anciennepageAhtml ou anciennepagezhtml provoquera une redirection 301 vers nouvellepage.html. Pour indiquer que le . doit être pris en compte en tant que tel (et non comme un signe faisant partie de la syntaxe PCRE / *Perl Compatible Regular Expression*), il faudra placer un antislash devant :

```
RewriteRule ^anciennepage\.html$
```

Ainsi, le signe . sera considéré comme un point de ponctuation. Des référenceurs mal intentionnés pourraient générer du DuplicateContent facilement avec des règles mal définies.

Le leading slash

Dans le cas d'une redirection ou d'une réécriture d'URL avec la directive *RewriteRule*, la gestion du premier slash des URL à réécrire prête souvent à confusion. En effet, on peut souvent voir des règles avec et sans ce premier slash.

Exemple :

```
RewriteRule ^contact /contact.php [L]
```

vs

```
RewriteRule ^/contact /contact.php [L]
```

On pourrait penser que le premier exemple fonctionne uniquement avec l'option RewriteBase / (qui indique le préfixe à appliquer sur toutes les URL à réécrire). Mais c'est faux. La première règle fonctionnera par défaut sur les versions d'Apache supérieures ou égales à la version 2.x, puisque le 1er slash de l'URL demandé est considéré comme étant déjà intégré à la règle. A l'inverse le deuxième exemple fonctionnera pour les versions d'apache 1.x uniquement.

Dans le doute, il est toujours possible d'utiliser le caractère spécial « ? » dans le cadre des expressions rationnelles PCRE, qui indiquera que le caractère

qui précède peut être présent 0 ou 1 fois :
RewriteRule ^/?contactp /contact.php [L]
= L'URL « /contact » peut contenir 0 ou 1 fois un slash

Caractères spéciaux

Pour rediriger des URL contenant des caractères encodés (http://en.wikipedia.org/wiki/Percent-encoding), exemple: /ma%20page.html avec %20 = espace), les directives de redirection *RedirectPermanent*, *Redirect* ou *RedirectMatch* seront capables de les gérer mais il sera nécessaire d'adapter la syntaxe des règles. Il arrive que des liens externes mal formatés contiennent des espaces dans les URL par exemple, les rediriger vous permettra de bénéficier de leur popularité dans une stratégie de *BrokenLinkBuilding* (chassse aux liens cassés).

URL encodées

Avec *RedirectPermanent*, la règle suivante ne fonctionnera pas :
RedirectPermanent /ma page.html /ma-page.html

En effet, l'espace dans cet exemple sert à définir la séparation entre l'URL à rediriger et l'URL redirigée. Dans ce cas, on utilisera des guillements pour encadrer l'URL à réécrire :
RedirectPermanent "/ma page.html" /ma-page.html.

Avec les directives de mod_alias, les caractéres spéciaux ne devront donc pas être encodés (pas de %), mais l'expression devra être encadrée avec des guillemets.

Concernant la directive *RewriteRule* de mod_rewrite, la gestion des caractères spéciaux est différente. Prenons une URL mal formatée : /bougie/ma bougie.html (avec un espace entre ma et bougie.html) ce qui donnera une fois l'URL encodée :
/bougie/ma%20bougie.html
Voici la règle de redirection adéquate :
RewriteRule ^bougie/ma\x20bougie\.html$ /bougie/ma-bougie.html [L,R=301]
Pour chaque caractère encodé d'une URL à rediriger avec une *RewriteRule*, il faudra remplacer le % par \x.

Rediriger vers une ancre

Il peut être utile d'effectuer une redirection 301 vers des parties spécifiques d'une page (ex: plusieurs pages rassemblées en une seule). Avec les directives de mod_alias, il suffira d'ajouter un #nomdelancre à la fin de l'URL de destination. Cependant, si vous avez besoin d'utiliser une *RewriteRule* (détections supplémentaires), le caractère # sera encodé en hexadecimal et provoquera une erreur 404.
Exemple :
RewriteRule ^livraison /faq#livraison [L,R=301]

Sur l'appel de http://monsite.com/livraison, on sera redirigé vers http://monsite.com/faq%23livraison et on obtiendra une erreur.

Pour éviter que les caractères spéciaux de l'URL réécrite ne soient encodés, on utilisera le drapeau Apache NE (*NoEscape*). Voici donc la règle modifiée pour rediriger vers /faq#livraison :
RewriteRule ^livraison /faq#livraison [L,R=301,NE]

Gestion de la chaîne de paramètres

La gestion de la chaîne de paramètres est l'un des principaux facteurs d'erreurs lors de la mise en place de redirections. En effet, lors d'une migration de site par exemple, c'est parfois une opportunité pour se débarrasser d'anciens paramètres de tri ou de tracking sur les URL. Mais bien souvent, ces redirections ne tiennent pas compte des chaînes de paramètres, qui se retrouvent présentes sur les URL du nouveau site. On obtient donc un transfert des paramètres d'un ancien CMS vers le nouveau, ce qui ne sera pas optimal pour l'indexation et le crawl du site par Googlebot, et génèrera du contenu dupliqué.

Suppression de la chaîne de paramètres

Avec la règle de redirection suivante :
RewriteRule ^actualites/international$ /news/international/ [L,R=301]
L'URL http://monsite.com/actualites/international?tri=desc&number=10 sera redirigée vers http://monsite.com/news/international/?tri=desc&number=10. Hors, le nouveau site n'utilise pas nécessairement les mêmes paramètres de tri, ou de tracking. Nous préférons donc une redirection vers http://monsite.com/news/international/ avec suppression de la chaîne de paramètres.

Pour cela, vous devrez obligatoirement passer par une *RewriteRule*, car les directives *Redirect, RedirectPermanent* et *RedirectMatch* ne prennent pas en charge les chaînes de paramètres.

- Pour les versions d'Apache inférieures à 2.4, on utilisera un point d'interrogation à la fin de l'URL réécrite pour supprimer la chaîne de paramètres :
RewriteRule ^actualites/international$ /news/international/? [L,R=301]

- Pour les versions d'Apache supérieures ou égales à 2.4, on utilisera le drapeau QSD (*Query String Discard*) pour supprimer la chaîne de paramètres :
RewriteRule ^actualites/international$ /news/international/ [L,R=301,QSD]

Conserver la chaîne de paramètres

La chaîne de paramètres est récupérée par défaut avec les règles de redirections, que l'on utilise une directive *RedirectPermanent, RedirectMatch* ou encore *RewriteRule*.

Avec la règle RewriteRule ^blog/$ /actus/ [L,R=301], si l'URL appelée est http://monsite.com/blog/?origin=abondance, l'internaute sera redirigé vers http://monsite.com/actus/?origin=abondance : la chaîne de paramètres sera recopiée par défaut. Le drapeau QSA qui s'utilise dans les *RewriteRule* et permet d'ajouter la chaîne de paramètres à l'URL réécrite n'a donc aucune utilité dans ce cas. Bien que la chaîne de paramètres soit ajoutée par défaut, on trouve parfois le drapeau QSA présent par erreur dans des règles de réécriture (ce qui provoque une chaine de paramètres doublée). Cependant, il devient fort utile quand l'URL réécrite contient elle-même une chaine de paramètres, un exemple sera plus parlant :
RewriteRule ^partenaires$ /index.php?page=partenaires [L]

Sur un appel de la page partenaires, Apache ira chercher le contenu de l'URL index.php? page=partenaires en back-end. Jusque-là, pas de problèmes. Cependant, si l'URL à réécrire contenait déjà une chaîne de paramètre (paramètre de tracking par exemple) que l'on souhaite conserver, elle ne sera pas ajoutée par défaut puisque nous en forçons déjà une dans l'URL réécrite avec « ?page=partenaires »

http://monsite.com/partenaires?affid=2345 appellera l'URL http://monsite.com/index.php?page=partenaires

C'est là que le drapeau Apache QSA a toute son utilité (QSA : *Query String Append*) :

```
RewriteRule ^partenaires$
/index.php?page=partenaires [L,QSA]
```

Grâce au drapeau QSA, http://monsite.com/partenaires?affid=2345 redirigera vers l'URL http://monsite.com/index.php?page=partenaires&affid=2345 en ajoutant les paramètres déjà présents en entrée de l'URL à réécrire.

Rediriger en fonction de la chaîne de paramètres

Vous pouvez devoir récupérer une partie de la chaîne de paramètres d'une URL ou avoir à effectuer des redirections en la présence de certaines variables dans la chaîne de paramètres. Vous serez dans ce cas obligé d'utiliser une règle de réécriture couplée à une ou plusieurs conditions de réécriture (*RewriteCond*) pour analyser le contenu des paramètres d'URL, variable %{QUERY_STRING} sur Apache.

L'utilisation des directives du module mod_alias (*Redirect, Redirect Permanent, RedirectMatch* par exemple) ne pourront pas répondre à vos besoins.

Exemple : Si nous avons une variable de pagination que l'on souhaite conserver dans l'URL des actualités d'un site, mais que le nom de cette variable est modifié (« p= » devient « page= »), voici la règle et sa condition de réécriture adéquate :

```
RewriteCond %{QUERY_STRING}
^p=([0-9]+)
RewriteRule ^actualites$
/news?page=%1 [L,R=301]
```

Explications : ([0-9]+) représente un chiffre de 0 à 9 répété une ou plusieurs fois (signe + qui signifie « une ou plusieurs occurrences du caractère précédent »). La variable %1 reprend le contenu du 1er ensemble défini dans la *RewriteCond* avec des parenthèses. Ainsi, /actualites?p=3 redirigera vers /news?page=3

Si la chaîne de paramètres (représentée par la variable %{QUERY_STRING} sur Apache) contient le paramètre "p" ayant une valeur numérique, alors on redirigera la page /actualites vers la page /news, en modifiant le paramètre « p » par « page » et en conservant sa valeur.

Règles de réécriture : de la rigueur avant tout

Vous l'aurez compris, la manipulation des URL sur un serveur Apache présente un fort potentiel, mais elle doit se faire avec rigueur pour éviter des baisses de performances en termes de temps de chargement, des dysfonctionnements en cas de règles trop permissives, ainsi que de la génération de Duplicate Content si l'on

ne tient pas compte des paramètres d'URL dans la mise en place de ces redirections ou réécritures.

Dans un prochain article, nous verrons comment manipuler les URL de façon plus complexe lors d'une migration SEO, tout en limitant les baisses de performance du serveur.

Source : http://httpd.apache.org/docs/

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