



La redirection HTTP

Par Spy-Seth



www.siteduzero.com

*Licence Creative Commons BY-NC 2.0
Dernière mise à jour le 20/08/2011*

Sommaire

Sommaire	2
La redirection HTTP	3
Introduction à la syntaxe	4
Changement permanent de nom de domaine	4
Déplacement définitif d'un dossier	5
Déplacement définitif d'un fichier	5
Suppression définitive d'un fichier	5
Changer l'extension d'un fichier	6
Redirection temporaire	7
Partager	8



La redirection HTTP

Par  Spy-Seth

Mise à jour : 04/05/2009

Difficulté : Facile 



357 visites depuis 7 jours, classé 246/782

Vous venez de déménager votre site ? De changer l'URL d'une / de plusieurs page(s) ? D'un dossier ? Mais malheureusement vous avez peur que vos visiteurs s'affolent s'ils ne trouvent plus la / les pages déplacée(s) ? Que les moteurs de recherche soient perdus ?

J'ai la solution : les redirections.

Je dis *les* et non *la* redirections, car il existe plusieurs méthodes de redirection :

- la redirection HTTP ;
- la redirection HTML ;
- la redirection PHP ;
- la redirection Javascript.

Comme vous vous en doutez, il y en a de meilleures que d'autres, et c'est pour cela que je vais essayer de toutes les aborder en montrant les points positifs et négatifs de chacune d'elles.

Tout cela se fera sur plusieurs tutoriels ; dans celui-ci, je me contenterai de vous expliquer la redirection HTTP qui est à mon avis la meilleure de toutes. En tout cas, c'est sûrement la plus utilisée.

Avant de commencer, il vous faut savoir une chose : cette redirection fonctionne directement avec Apache.





Pour utiliser cette fonction, il vous faudra un serveur qui tourne sous Apache et que sa configuration accepte la directive `Redirect` dans les fichiers `.htaccess`.

La majeure partie des serveurs tournent sous Apache, l'autre partie (qui ne tourne pas sous Apache) comprend néanmoins les mêmes fichiers que ceux utilisés sur des serveurs Apache : ça ne pose donc pas de problème, généralement.



`.htaccess` : mais je connais ça, ce n'est pas le fichier dans lequel on configure les erreurs 40x (400, 401, 402, 403, 404...) ? 

Très bonne remarque ! C'est effectivement dans le même fichier que l'on fait ces deux manipulations. 

Pour commencer, je vais vous donner un fichier d'exemple qui va vous montrer les possibilités de la chose. 

Code : Apache

```
Redirect permanent /dossier01/page01.html
http://nouvelle.adresse.fr/dossier03/page02.html
RedirectMatch seeother ^/images/(.*)\.jpg$
http://adresse.actuelle.fr/images/$1.gif
Redirect permanent /dossier01 http://nouvelle.adresse.fr/dossier02
Redirect permanent / http://nouvelle.adresse.fr/
Redirect gone /vieux.html
```



Oullaaaaaalaalaaa... c'est quoi, ce truc ? 🤔🤔

Ça, c'est un fichier .htaccess avec des règles de redirections complexes (les expressions régulières), mais pas de panique : je ne vais rien vous faire faire d'aussi compliqué !

Sommaire du tutoriel :



- Introduction à la syntaxe
- Changement permanent de nom de domaine
- Déplacement définitif d'un dossier
- Déplacement définitif d'un fichier
- Suppression définitive d'un fichier
- Changer l'extension d'un fichier
- Redirection temporaire

Introduction à la syntaxe



Les fichiers .htaccess ont une syntaxe particulière, elle n'est pas dure mais il faut la connaître. 😊

Chaque ligne d'un fichier .htaccess est une instruction.

On peut décomposer les lignes en quatre parties :

Code : Apache

```
RedirectMatch seeother ^/images/(.*)\.jpg$  
http://adresse.actuelle.fr/images/$1.gif
```

Ici nous avons :

- RedirectMatch : c'est la directive, ou pour la redirection : l'instruction de redirection (ça dit ce que l'on va faire) ;
- seeother : c'est une option de la directive RedirectMatch ;
- ^/images/(.*)\.jpg\$: la cible de l'instruction de redirection ;
- http://adresse.actuelle.fr/images/\$1.gif : l'URL d'arrivée (dans notre cas : l'URL vers laquelle on redirige).

Pour plus de documentation sur les différentes directives utilisables : voir [la documentation officielle des alias d'Apache \(fr\)](#) ou [la documentation officielle sur les requêtes HTTP 1.1 \(en\)](#).

Changement permanent de nom de domaine

Le changement d'adresse est toujours risqué : il a des risques que le site coule suite à un non-suivi des visiteurs et / ou un *déréfèrencement* du site.

On peut palier ce problème avec un type de redirection HTTP très précise.

Voilà comment ça se formule :

Code : Apache

```
Redirect permanent / http://nouvelle.adresse.fr/
```

Prenons l'exemple sur d'un site personnel qui est hébergé chez free (<http://monsite.free.fr>) : le webmaster décide de s'acheter un nom de domaine et un hébergement : <http://www.monsite.sdz>. Il copie donc tout le site de l'ancienne vers la nouvelle adresse, mais pour conserver son référencement, il a mis en place une redirection HTTP du type de celui que nous venons de voir.

Il voulait que l'adresse <http://monsite.free.fr> redirige vers <http://www.monsite.sdz>, il a donc mis un fichier .htaccess à la racine de <http://monsite.free.fr> qui contenait :

Code : Apache

```
Redirect permanent / http://www.monsite.sdz/
```

Explication du code :

- Redirect permanent spécifie le type de redirection ;
- / après *Redirect permanent* dit qu'il faut rediriger toutes les pages ;
- <http://www.monsite.sdz/> est l'adresse de destination de la redirection.

Le gros avantage de cette redirection est que si le visiteur demande la page : <http://monsite.free.fr/hot-dog.html>, il sera automatiquement redirigé vers <http://www.monsite.sdz/hot-dog.html>. 😊



C'est bon, vous suivez ? S'il y a un problème, relisez ce passage. 😊 Si en ayant relu plusieurs fois vous ne comprenez toujours pas, passez à la suite, ça rentrera sûrement avec d'autres exemples. ;-)

Déplacement définitif d'un dossier

C'est le même principe que pour changer l'adresse d'un site, sauf que l'on remplace le sélecteur "/" (qui voulait dire "tout") par le chemin du dossier. 😊

Ce qui nous donne pour déplacer <http://www.monsite.com/dossier1/> vers <http://www.monsite.com/dossier2/> :

Code : Apache

```
Redirect permanent /dossier1 http://www.monsite.com/dossier2/
```



Qui a dit que c'était compliqué ? 😊

Déplacement définitif d'un fichier

Lorsque vous renommez un fichier de votre site, il est souvent utile de mettre une redirection à l'ancien emplacement vers le nouvel emplacement. Pour ce faire, on utilise la même directive de redirection que pour déplacer un dossier : Redirect permanent ; mais on cible un fichier au lieu de cibler un dossier.

Par exemple, si je veux déplacer <http://www.monsite.com/monfichier.html> vers <http://www.monsite.com/mondossier/monfichier.html>, cela donne :

Code : Apache

```
Redirect permanent /monfichier.html  
http://www.monsite.com/mondossier/monfichier.html
```

Suppression définitive d'un fichier

Cette technique est utilisée pour indiquer aux visiteurs et aux moteurs de recherche que le fichier n'existe plus (et qu'il ne renaîtra

pas) ; par conséquent, ils peuvent se mettre à jour. Cela se fait par l'intermédiaire d'une erreur *410 document n'existe plus*.

Ici, on utilise qu'une seule adresse : celle du fichier supprimé (ici : `/chemin/de_fichier_supprime.html`).
Et comme option de directive, `gone`. Ce qui donne :

Code : Apache

```
Redirect gone /chemin/du_fichier_supprime.html
```

Changer l'extension d'un fichier

Le changement d'extension de fichier est sûrement la redirection la plus utilisée. Par exemple sur le SdZ, pour toutes les pages ; si vous regardez l'adresse de ce tutoriel : [http://www.siteduzero.com/tuto-3-4963- \[...\] ion-http.html](http://www.siteduzero.com/tuto-3-4963- [...] ion-http.html), en réalité cette page n'est pas un fichier HTML, mais un fichier PHP du genre `/tuto.php?variable1=3&variable2=4963&variable3=1`.



Ce n'est pas réellement comme ça que s'appelle le fichier `.php`, mais c'est un fichier dans le même genre (*M@teo, corrige-moi si je vais droit dans le mur* 😊).



Il faut que je signale que cette instruction est souvent bloquée par les hébergeurs car elle est assez gourmande en ressources, et qu'il est très facile de se tromper dans l'écriture des redirections (ce qui entraîne une surcharge du serveur) ! Vérifiez qu'elle est disponible avec la fonction `phpinfo()` de PHP.

Revenons à nos moutons.



Pourquoi cacher le nom original d'un fichier ?

Pour deux raisons :

- le référencement : les moteurs de recherche n'aiment pas trop les noms de fichiers PHP avec des variables, c'est ce qui est fait ici ;
- la génération d'image en PHP : si l'on veut intégrer une image générée par PHP dans un post d'un forum, ou dans une page XHTML, il faut qu'elle ait une extension d'image (c'est-à-dire : `.jpg`, `.jpeg`, `.gif`, `.png`...), et cette technique permet de cacher le nom de l'image.

Prenons pour exemple le deuxième cas : j'ai créé un fichier `pseudo.php` qui génère un texte en image ; ce texte est contenu dans la variable `$texte` qui est récupéré via `$GET`, ce qui donne un fichier `pseudo.php?texte=sdz`.

Ce fichier-là ne peut être affiché dans les forums car il faut afficher une image `.jpg`, `.jpeg`, `.gif`, `.png`. Grâce au changement d'extension de fichier, je peux faire croire aux forums que le fichier `pseudo.php?texte=SDZ` est une image `.png` (ou autre). Pour ce faire, je crée un fichier `.htaccess` avec comme instruction :

Code : Apache

```
Redirect seeother /pseudo-texte-sdz.png  
http://www.monsite.com/pseudo.php?texte=sdz
```

Donc là, si je demande le fichier `www.monsite.com/pseudo-texte-sdz.png`, c'est comme si je demandais le fichier `www.monsite.com/pseudo.php?texte=SDZ`.



Si on utilise `$get` pour récupérer la variable `$texte`, c'est pour pouvoir changer de texte facilement, non ? Et là, avec cette technique, c'est impossible.

Oui, vous avez raison : si on utilise `$get`, c'est bien pour pouvoir changer facilement le texte généré. Et en utilisant le code précédent, cela est impossible, c'est pour ça que les expressions régulières existent. Cela permet de faire passer une variable de l'URL cible, à l'URL d'arrivée.

Je vais vous expliquer avec un exemple pratique. Reprenons le code précédent : nous avons donc dans l'URL d'arrivée la variable `$texte` qui a pour valeur "SDZ". si je veux changer sa valeur? il suffit de changer le contenu de `$texte` dans l'URL, par exemple : `pseudo.php?texte=prout`, ce qui donnerait www.monsite.com/pseudo-texte-prout.png. Pour arriver à ce résultat, il faut utiliser les expressions régulières. Ce qui donne :

Code : Apache

```
RewriteRule ^pseudo-texte-([^\/]*)\.png$ pseudo.php?texte=$1 [L]
```

Si vous souhaitez plus d'informations sur les expressions régulières et cette redirection (elle est poussée, il y a des livres entiers à son sujet), je vous conseille, en plus des liens donnés au début de ce tutoriel, le tutoriel écrit à ce sujet pour le CMS SPIP (mais qui convient très bien à une utilisation hors SPIP) : [La réécriture des URL "à la volée"](#).

Redirection temporaire

Les plus malins d'entre vous auront remarqué que pour toutes les redirections que j'ai présentées, j'ai utilisé les mots *définitif* et *permanent* : cela n'est pas pour faire joli, mais parce qu'il y a des redirections temporaires.



Temporaire ? Pour quoi faire ? 🤔

Prenons un exemple pour expliquer : vous réalisez des travaux dans une partie de votre site, et vous désirez de fermer cette partie au public pour la figoler. Pour être tranquilles, le mieux est d'utiliser une redirection temporaire.



Mais pourquoi s'embêter à utiliser une redirection temporaire alors que les autres feraient l'affaire ?

Il est vrai que moi aussi je me suis posé la question à ce sujet-là, et j'ai eu la chance d'avoir une réponse très rapide auprès d'amis beaucoup plus compétents que moi 😊.

Donc, si vous mettez une redirection permanente et qu'un bot de référencement vient pendant qu'il y a la redirection, il va mettre à jour sa base de données et par conséquent, faire perdre le référencement de cette partie 😞 ; alors que si vous utilisez une redirection temporaire, le moteur ne mettra pas à jour sa base de données et surtout, vous conservera votre référencement 😊.

Maintenant que l'on sait à quoi ça sert, il faudrait savoir comment on s'en sert. 😊

C'est toujours aussi simple que les autres : on utilise `Redirect` comme directive, `temp` comme option de directive et les adresses (la cible et l'URL d'arrivée) ; dans notre exemple, nous utiliserons `/zone_en_finition/` comme cible et `/zone_temporaire/` comme URL d'arrivée.

Code : Apache

```
Redirect temp /zone_en_finition/  
http://www.monsite.com/zone_temporaire/
```



Alors, c'est dur ?

Voilà pour ce qui est de la redirection HTTP : c'est très complet et normalement, avec ça et les expressions régulières, vous devriez pouvoir rediriger un bon nombre de personnes.

Si vous avez un problème, n'hésitez pas à demander de l'aide. ;-)

Partager



Ce tutoriel a été corrigé par les [zCorrecteurs](#).