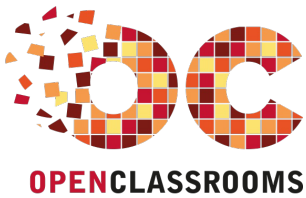
Du rewriting réalisé avec du PHP

Par Jérémy_B



www.openclassrooms.com

Sommaire

Du rewriting réalisé avec du PHP

Par  Jérémy_B

Mise à jour : 01/01/1970
Difficulté : Intermédiaire

L'URL Rewriting est souvent nécessaire pour les sites de grande envergure, tout simplement pour masquer le nom des paramètres envoyés dans les URL, pour rendre les adresses plus séduisantes, mais aussi pour accroître le référencement (mais pas dans tous les cas). Vous aurez sans doute remarqué que les hébergeurs ne proposent pas tous l'*URL Rewriting*, c'est le cas de Free. Les choses sont révolues maintenant, nous pouvons aussi, à présent, créer de l'*URL Rewriting* avec un peu de tricherie, mais intelligemment.

Vous avez toujours rêvé d'avoir de beaux liens comme celui-ci : http://monsite.fr/page-1.html, ou encore http://monsite.fr/forum-1-maphraseavec-des-tirets.html ?
Vous en avez marre d'avoir ce type de lien : http://monsite.fr/page.php?id=1 ?
Ou encore ceci : http://monsite.fr/?/page-1.html ?

Alors suivez le guide !

Liens utiles :

- le cours de php de M@teo21, mais plus particulièrement les expressions régulières
- explication de l'URL Rewriting sur Webrankinfo.com
- explication de l'URL Rewriting sur Urlrewriting.fr
- comment définir correctement des URL (à partir de la section : définir les schémas d'URL)
- du Rewriting sans le moteur rewrite

Dans ce tutoriel, je ne vous apprends pas l'astuce qui permet de créer ce type d'URL, car tout est indiqué dans le dernier lien utile, mais plutôt comment s'en servir de façon plus rigoureuse avec des exemples et des astuces.

Sommaire du tutoriel :

Utilisation pour une news

Pour comprendre son utilisation, rien de mieux que de l'expliquer en donnant un exemple concret. Pour cela, nous allons nous intéresser au système de news. J'ai choisi cet exemple, tout simplement parce que je suis sûr que vous aurez déjà vu comment cela fonctionne, si vous avez lu le tutoriel de M@teo21.

Pour éviter de trop se prendre la tête avec les URL, les *includes*, etc., nous commencerons par mettre nos fichiers à la racine du FTP.

Code pour la news

Code : SQL - La base de données

```
CREATE TABLE IF NOT EXISTS `news` (
  `id` mediumint(9) NOT NULL auto_increment,
  `auteur` varchar(50) NOT NULL,
  `titre` varchar(200) NOT NULL,
  `message` text NOT NULL,
  `temps` datetime NOT NULL,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=4 ;

INSERT INTO `news` (`id`, `auteur`, `titre`, `message`, `temps`)
VALUES
(1, 'Jérémy_B', 'L''URL rewriting', 'C''était pas si dur que ça !',
'2008-12-31 01:36:14'),
(2, 'Jean', 'Re !', 'C''est vrai que c''est tout simple.', '2009-01-
10 12:46:24'),
(3, 'Paul', 'Re re !', 'Merci Jérémy_B. :)', '2009-01-10 13:39:13');
```

Code : PHP - Le fichier news.php

```
<?php

 mysql_connect('localhost', 'root', '');
 mysql_select_db('urlrewriting'); // Mettez ici le nom de votre
base de données.

?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" >
 <head>
  <title>Page de news avec URL Rewriting !</title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-
8859-1" />
 </head>
 <body>

<?php

$requete = "SELECT id, auteur, titre, message, DATE_FORMAT(temps,
'%d/%m/%Y') AS temps_formate FROM news";

if (!empty($match)) {
 $requete .= " WHERE id = ".$match[1];
}

$resultat = mysql_query($requete);

while ($donnees = mysql_fetch_assoc($resultat)) {
 echo "Id : ".$donnees['id']."<br />";
 echo "Auteur : ".stripslashes($donnees['auteur'])."<br />";
 echo "Titre : ".stripslashes($donnees['titre'])."<br />";
 echo "Message : ".stripslashes($donnees['message'])."<br />";
 echo "Temps : ".$donnees['temps_formate']."<br />";
}

?>

 </body>
</html>
```

Voilà, jusque-là, rien de compliqué. Si vous avez été attentifs, vous aurez vu une petite condition vérifiant si la variable $match existe, et si elle n'est pas nulle. Elle nous sera utile lorsqu'on aura mis en place les codes pour l'*URL Rewriting*. D'ailleurs, il est

temps de passer aux pages qui nous intéressent le plus.

Code pour l'URL Rewriting

Code : Autre - Le fichier .htaccess

```
php 1
ErrorDocument 404 /rewriting.php
```

Explication

- *php 1* : ça n'a aucun rapport avec l'*URL Rewriting*, à part activer PHP 5 chez Free. Donc, si vous n'êtes pas chez cet hébergeur, ne mettez pas cette ligne dans votre ".htaccess".
- *ErrorDocument 404 / rewriting.php* : cette ligne permet de créer une petite personnalisation lorsque vous avez une erreur 404. En gros, au lieu de voir la page d'erreur générée par PHP, on verra votre page rewriting.php. Les plus malins d'entre vous auront compris que c'est avec cette page qu'on va un peu tricher.
  Note : vous n'êtes pas obligés d'appeler votre page rewriting.php, vous pouvez l'appeler, par exemple, urlrewriting.php ou autre.

Code : PHP - Le fichier rewriting.php

```
<?php

if (preg_match('#news-([0-9-]+)\.html#isU',
$_SERVER['REDIRECT_URL'], $match)) {
 // Modification du code retour, pour que les moteurs de recherche
indexent nos pages !
 header("Status: 200 OK", false, 200);

 require 'news.php';
}
else {
 require 'erreur404.php';
}
```

Explication

- *$_SERVER['REDIRECT_URL']* : c'est une variable prédéfinie créée par PHP, qui nous permet d'obtenir l'URL de la page souhaitée.
  Note : il se peut que votre hébergeur n'ait pas activé toutes les variables $_SERVER dont celle-ci. Vérifiez, ainsi, dans le PHP Info > PHP Variables, qu'elle est bien activée.
- *preg_match('#news-([0-9-]+)\.html#isU', $_SERVER['REDIRECT_URL'], $match)* : un exemple pour comprendre ce code. Soit http://monsite.fr/news-1.html, la page qu'on souhaite afficher (donc, $_SERVER['REDIRECT_URL'] = /news-1.html). Avec cette fonction, on peut récupérer le chiffre 1, grâce à l'expression régulière ([0-9]+) (ou appeler plutôt class), qui sera stocké dans la variable $match.

$match n'est pas qu'une simple variable, mais un tableau (à noter que $match[0] vaut : /news-1.html). Par exemple, si vous voulez transférer deux paramètres dans l'URL, votre expression régulière devrait être : "^/news-([0-9]+)-([0-9]+)\.html$", vous pourrez les récupérer avec $match[1] et $match[2]. Vous pouvez aussi récupérer autre chose que des chiffres, comme du texte. Par exemple, avec http://monsite.php/news-maphraseavec-des-tirets.html, votre REGEX sera : "^/news-([a-zA-Z_-]+)\.html$" et $match[1] vaudra : maphraseavec-des-tirets.

Il faut absolument que le paramètre contenant du texte dans l'URL soit accroché par quelque chose (lettre collée, tiret, etc.), mais surtout pas d'espace !

Déroulement des évènements

On va donc faire appel à la page de news, pour afficher la toute première actualité. Ainsi, rentrons dans la barre d'adresse de notre navigateur, l'adresse suivante : http://monsite.fr/news-1.html.

Mais cette page n'existe pas sur notre FTP... ça va me retourner une erreur, non ?

Exact, et pas n'importe quelle erreur, la 404. C'est là que notre tricherie va opérer ! Je vous ai dit qu'avec le fichier .htaccess, on allait rediriger l'erreur sur la page rewriting.php. Avec notre page de code, on fait une condition sur l'URL qu'on nous a envoyée, on récupère le chiffre 1, nous permettant, en réalité, de prendre l'id de la news, on renvoie un *header* pour dire que ce n'était pas une erreur, puis on inclut la page news.php, qui elle, existe réellement.

Je vous rappelle qu'on a récupéré l'id dans la variable $match[1]. Je vous rappelle aussi qu'une variable peut être transmise dans une autre page qu'on inclut juste après sa déclaration, à condition que dans la deuxième page, la variable ait le même nom.

Pour en revenir au déroulement des opérations, on vient donc d'inclure la page news.php. Sur cette page, notre requête peut avoir deux cas de figure :

- soit il y a un paramètre, l'id de la news, qui vaudra 1 pour nous. La condition sera donc validée et nous concaténerons la variable $requete grâce au ".=", afin d'ajouter un critère de sélection à notre requête ;
- soit notre URL n'avait pas de paramètre, on ignore alors la condition, car $match sera vide et n'existera tout simplement pas.

Pas la peine de protéger notre variable $match[1] dans la requête avec la fonction intval() ou autre de PHP, puisqu'elle a déjà eu le droit à un petit test avec la REGEX (([0-9]+), vous vous en souvenez ?). Nous sommes sûrs d'avoir affaire à un chiffre, donc pas d'inquiétude, nous n'aurons pas d'injection SQL ici.

Et voilà, miracle, ça marche ! Qu'on appelle notre page avec l'adresse http://monsite.fr/news-1.html ou avec http://monsite.fr/news.html, on visualise bien notre news.

Améliorations et astuces

Pour que notre script soit le plus intéressant possible, il faudrait qu'on puisse avoir plusieurs possibilités dans nos URL. Mais notre script a ses limites, on ne peut pas tout faire. Nous devrons donc mettre en place une cohérence dans nos liens, sinon, on ne s'en sortira jamais. Ainsi, dans les paramètres de l'URL, nous mettrons, dans un premier temps, les chiffres (id de la news, du membre, du commentaire, du topic, etc.), puis, dans un second temps, les chaînes de caractères. Nous pourrons donc arriver à ce type de lien :

- http://monsite.fr/page.html
- http://monsite.fr/page-1.html
- http://monsite.fr/page-1-2.html
- http://monsite.fr/page-1-le-super-titre.html
- http://monsite.fr/page-1-2-le-2-em-super-titre.html (la chaîne de caractères peut contenir des chiffres)
- etc.

Code : PHP - Le fichier rewriting.php

```php
<?php

$page_du_site = array("/news"); // Super important les / devant le
nom du fichier !
$url_sans_parametre = str_replace('.html', '',
$_SERVER['REDIRECT_URL']);
$url_avec_parametre = substr($_SERVER['REDIRECT_URL'], 0,
```

```
strpos($_SERVER['REDIRECT_URL'], "-"));

if (in_array($url_sans_parametre, $page_du_site)) {
 $url = dirname(__FILE__).$url_sans_parametre;
}
elseif (in_array($url_avec_parametre, $page_du_site)) {
 $url = dirname(__FILE__).$url_avec_parametre;
}
else {
 $url = 'erreur404';
}

$page = basename($url_avec_parametre);

if (preg_match('#'.$page.'-([0-9-]+)-([^0-9](?:.+))\.html#isU',
$_SERVER['REDIRECT_URL'], $match)) {
 header("Status: 200 OK", false, 200);
}
elseif (preg_match('#'.$page.'-([0-9-]+)\.html#isU',
$_SERVER['REDIRECT_URL'], $match)) {
 header("Status: 200 OK", false, 200);
}
else {
 header("Status: 200 OK", false, 200);
}

require $url.'.php';
```

Explication

- *$page_du_site = array("/news");* : on cherche dans le tableau $page_du_site, si notre URL correspond à une de ces valeurs. Si elle existe, on crée une URL valide, pour la future inclusion, sinon, on indique la page pour gérer, cette fois-ci, la véritable erreur 404. Rien de bien compliqué en soit.
- *str_replace('.html', '', $_SERVER['REDIRECT_URL']);* : ce code nous permet d'éliminer tout ce qui ne nous convient pas dans la variable $_SERVER. En effet, si on souhaite juste appeler la page /news.html, on souhaiterait avoir simplement le mot "/news" pour pouvoir comparer avec d'autres mots.
- *substr($_SERVER['REDIRECT_URL'], 0, strpos($_SERVER['REDIRECT_URL'], "-"));* : si nous avions l'URL suivante : http://monsite.fr/news-1.html, le code nous permettrait de ne récupérer que "/news". Utile, puisqu'on va pouvoir le comparer avec les valeurs du tableau $page_du_site.
- *-([^0-9](?:.+))* : dans les deux conditions, la seule différence se situe avec cette classe, qui permet de récupérer une chaîne qui ne commence pas par un chiffre.
- *require $url.'.php';* : on inclut la page qui nous intéresse, d'où l'intérêt de nos trois premières lignes de code. Ainsi, avec n'importe quelle page de notre FTP, on peut faire de l'*URL Rewriting*.
- Ainsi, le code nous permet de vérifier l'existence de la page sur notre FTP, auquel cas on le renvoie sur l'erreur 404.php. Ensuite, on fait différentes conditions pour savoir si l'URL a des paramètres (chiffre, ou chiffre et lettre) ou non. Et dans tous les cas, on indique, grâce au *header*, que tout se passe bien, qu'il y avait un malentendu  . Puis on inclut la page que l'on voulait.

N'oubliez pas que si vous voulez faire de l'*URL Rewriting*, sans passer d'argument (ou de paramètre), il faut que le nom de la page soit aussi dans le tableau, sinon, pas d'*include* (enfin si, l'erreur 404) !

Maintenant que nous avons mis le code en place, créez la page page.php, permettant de bien comprendre ce que fait notre script du dessus. Voici le code :

Code : PHP - Le fichier page.php

```
<?php

$taille = count($match);

for ($i=0; $i<$taille; $i++) {
```

```
        echo 'match['.$i.'] vaut : '.$match[$i].'<br />';
    }
```

Si vous utilisez cette page pour faire vos tests, n'oubliez pas qu'il faut rajouter le nom de cette page dans le tableau $page_du_site !

Il ne vous reste plus qu'à tester les différentes URL possibles, et vous comprendrez bien mieux ce qu'on a voulu faire. Pour que tout le monde ait le même résultat, testez les URL que je vous ai données au début de ce sous-chapitre. Par exemple, pour http://monsite.fr/page-1-le-super-titre.html, on obtient ceci :

Code : Autre

```
match[0] vaut : page-1-2-le-super-titre.html <--
> correspond aussi à $_SERVER['REDIRECT_URL']
match[1] vaut : 1-2
match[2] vaut : le-super-titre
```

Mais... match[1] vaut 1-2 ! C'est nul ton truc ! Comment on fait si je ne voulais que le 1 ou le 2 ?

Pas de panique, on y va tranquillement pour bien comprendre. Pour séparer une chaîne en tableau avec PHP, il faut utiliser la fonction explode. Nous allons donc passer $match[1] dans *explode*, et nous pourrons ainsi récupérer nos paramètres 1 et 2.

Code : PHP - Dans le fichier rewriting.php

```php
<?php

$parametre = explode('-', $match[1]);
print_r($parametre); // Array ( [0] => 1 [1] => 2 )
```

Et voilà, nous pouvons maintenant avoir nos beaux paramètres dans des variables : $parametre[0] = 1 et $parametre[1] = 2. Le code ci-dessus (sans le print_r()), est à placer, bien évidemment, dans le fichier rewriting.php, au niveau des deux conditions utilisant preg_match(). Il ne nous reste plus qu'à faire la même chose pour $match[2], qui contient, cette fois-ci, une chaîne. Sauf que là, on veut juste remplacer les tirets (-) par des espaces et, pourquoi pas, mettre une lettre majuscule pour la première lettre !

Code : PHP

```php
<?php

$titre = ucfirst(str_replace('-', ' ', $match[2]));
echo $titre; // Le super titre
```

On peut aussi s'aider du CSS, si on le souhaite pour la première lettre, d'après le cours de M@teo21 !

Nous avons vu comment avoir les bons paramètres dans plusieurs variables pour nous faciliter la tâche, mais nous n'avons pas vu comment créer ces liens ! Eh oui, car bon, pour les chiffres, il n'y a pas trop de problèmes, mais pour les chaînes de caractères... c'est une tout autre paire de manches. On a, facilement, la possibilité de faire des URL invalides... et ce n'est pas notre but. Bref, voyons comment régler ce problème.

Tout d'abord, reprenons l'exemple de notre news. On va, cette fois-ci, sélectionner une news avec tous ses champs, et ainsi, on pourra, avec son id et son titre, créer un lien pour notre belle URL :

Code : PHP - Le fichier news.php

```
<?php

 mysql_connect('localhost', 'root', '');
 mysql_select_db('urlrewriting'); // Mettez ici le nom de votre
base de données

 function caractereValideUrl($string) {
  $string= strtr($string,
   "ÀÁÂÃÄÅàáâãäåÒÓÔÕÖØòóôõöøÈÉÊËèéêëÇçÌÍÎÏìíîïÙÚÛÜùúûüÿÑñ",
   "aaaaaaaaaaaaooooooooooooeeeeeeeecciiiiiiiiuuuuuuuuynn");

  // Mettez ici les caractères spéciaux qui seraient susceptibles
d'apparaître dans les titres. La liste ci-dessous est indicative.
  $speciaux = array("?
","!","@","#","%","&amp;","*","(",")","[","]","=","+","
",";",":","'",".","_");
  $string = str_replace($speciaux, "-", $string); // Les caractères
spéciaux dont les espaces, sont remplacés par un tiret.
  $string = strtolower(strip_tags($string));

  return $string;
 }

?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" >
 <head>
  <title>Les news !</title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-
8859-1" />
 </head>
 <body>

<?php

$requete = "SELECT id, auteur, titre, message, DATE_FORMAT(temps,
'%d/%m/%Y') AS temps_formate FROM news";

if (!empty($match)) {
 $requete .= " WHERE id = ".$match[1];
}

$resultat = mysql_query($requete);

while ($donnees = mysql_fetch_assoc($resultat)) {
 // Tout ce qu'on souhaite afficher pour la news, comme au tout
début...
 echo '<a href="http://monsite.fr/news-'.$donnees['id'].'-
'.caractereValideUrl('Jérémy B'))))).'.html">Un superbe lien !</a>';
 // http://monsite.fr/news-1-jeremy-b.html
}

?>

 </body>
</html>
```

Explication

- *function caractereValideUrl($string)* : c'est une petite fonction nécessaire pour rendre l'URL plus "valide".
- *strtr(...)* : cette fonction nous permet de transformer les occurrences de chaque caractère accentué en caractère... sans accent.
- *str_replace(...)* : on remplace, tout simplement, les caractères spéciaux, qui peuvent rendre invalide l'URL, par des tirets.
- *strip_tags(...)* : strip_tags permet de supprimer d'éventuelles insertions malveillantes de code HTML ou PHP, dans le cas où le formulaire est public.
- *strtolower(...)* : un lien ou une URL peut contenir des majuscules, mais avec cette petite fonction bien pratique, on va faire une URL plus propre.

En réalité, $match[2] ne nous sert finalement à rien... Pourquoi ? Tout simplement parce qu'on ne va pas s'amuser à faire la conversion, alors qu'on ne saura pas s'il y avait bien un accent, une apostrophe, etc. Autant le sélectionner directement dans la base de données. Son intérêt est de faire des liens plus explicites, qui nous parle.

Et une dernière amélioration pour la route : tout ceci se passe à la racine du FTP, mais qu'en est-il pour les autres fichiers qui se trouvent dans des dossiers ? Comme à notre habitude, on va devoir modifier la page qui nous intéresse : rewriting.php. La solution est toute simple, il suffit de mettre le nom du dossier avec la page dans le tableau $page_du_site.

Code : PHP - Tableau dans le fichier rewriting.php

```php
<?php

$page_du_site = array("/test/page", "/news");
```

Bon, je crois qu'on a fait le tour des améliorations possibles. Comme quoi, avec notre script, nous avions au départ un code tout simple, nous l'avons complexifié, en rajoutant diverses améliorations, et nous avons fait en sorte d'avoir pensé à toutes les éventualités.
Voilà, le tutoriel touche à sa fin. Vous savez maintenant qu'on peut faire du *Rewriting*, sans pour autant avoir le moteur adéquat dans Apache (désactivé par Free, par exemple). Ne me dites pas, par contre, que ça ne marche pas ou autre, car j'ai bien tout testé avant d'avoir écrit ce tutoriel.

Le tutoriel n'est sûrement pas parfait, il doit bien y avoir quelques erreurs, mais il n'est cependant pas figé. Venez donc m'en faire part pour que je puisse l'améliorer !

La seule chose qui me manque comme information, ce serait de savoir si cette méthode est plus lente que celle du véritable moteur. Je n'ai, hélas, pas les connaissances suffisantes, mais bon, vu que notre système et celui du moteur utilisent tous les deux les REGEX, il ne doit pas y avoir d'énorme différence (de temps, bien sûr).

Merci, en tout cas, d'avoir lu jusqu'au bout, et j'espère en avoir aidé plus d'un !

Je remercie aussi au passage ptipilou pour sa Zcorrection.

Ce tutoriel a été corrigé par les zCorrecteurs.