PHP - Fichiers et dossiers

Date de dernière mise à jour : 27/06/2007 à 19:36

Présentation

Les fichiers et dossiers sont très souvent utilisés en PHP. Nous verrons plus loin qu'il existe d'autres manières de stocker les données en utilisant ce qu'on appelle une base de données, qui est une autre forme de stockage très couramment utilisée.

Nous allons apprendre pour ce chapitre à manipuler des fichiers existants (ouvrir un fichier, enregistrer des données et fermer le fichier).

Ouvrir un fichier en PHP

Nous supposerons que vous avez déjà un fichier qui se nomme "test.txt" (fichier texte donc) sur votre disque dur, dans le répertoire courant que vous utilisez pour travailler en PHP. Vous savez ce qu'est une fonction, donc je ne vais pas mettre plus de temps à vous dire que pour ouvrir un fichier, plusieurs fonctions sont disponibles en fonction de ce que l'on souhaite faire.

La plus simple est la fonction **file_get_contents()**. Elle prend comme paramètre le nom du fichier (nous ne détaillerons pas les autres paramètres possibles de cette fonction ici, ils ne nous seront pas utiles pour la suite). On récupère le contenu du fichier ouvert dans une variable. Voici comment ça se passe :

```
<?php     $fichier
=
file_get_contents
(
'test.txt'
);
//récupère le contenu du fichier et le place dans la variable $fichier
echo
$fichier
;
//affiche le contenu du fichier
?>
```

Vous n'avez pas à fermer le fichier quand vous utilisez cette fonction, qui à mon avis est adaptée quand vous ne souhaitez pas lire les données du fichier d'une certaine façon. Vous verrez qu'avec une autre fonction que nous allons voir maintenant, vous allez pouvoir faire plus de choses. La fonction que nous allons utiliser porte le doux nom de fopen() et demande au minimum deux paramètres devant être renseignés. Le premier paramètre est le nom du fichier (ce qui ne change pour le moment pas avec la fonction file_get_contents()), et le deuxième est un peu particulier. En effet, ce deuxième paramètre va être utile pour déterminer le mode d'ouverture du fichier. Voici un récapitulatif des modes disponibles :

mode
Description

r
Ouvre le fichier en lecture seule, le pointeur de fichier est placé au début du fichier.

r+
Ouvre le fichier en lecture + écriture, le pointeur de fichier est placé au début du fichier.

w
Ouvre le fichier en écriture seule; le pointeur de fichier est placé au début du fichier et écrase les données du fichier existant (si il existe). Si le fichier n'existe pas, ce mode permet de créer le fichier.

w+
Ouvre le fichier en lecture et écriture; le pointeur de fichier est placé au début du fichier et écrase les données du fichier existant (si il existe). Si le fichier n'existe pas, ce mode permet de créer le fichier.

a
Ouvre le fichier en écriture seule; place le pointeur de fichier à la fin du fichier. Si le fichier n'existe pas, il est créé.

a+
Ouvre le fichier en lecture et écriture; le pointeur de fichier est placé à la fin du fichier. Si le fichier n'existe pas, il est créé.

x
Crée et ouvre le fichier en lecture seule; le pointeur de fichier est placé au début du fichier. Si le fichier existe déjà, **fopen()** va échouer. Si le fichier n'existe pas, **fopen()** tente de le créer. Cette option est supportée à partir de PHP 4.3.2 et fonctionne uniquement avec des fichiers locaux.

x+
Crée et ouvre le fichier en lecture et écriture; place le pointeur de fichier au début du fichier. Si le fichier existe déjà, **fopen()** va échouer. Si le fichier n'existe pas, il est crée. Cette option est supportée à partir de PHP 4.3.2, et fonctionne uniquement avec des fichiers locaux.

Vous vous demandez peut-être ce qu'est un pointeur de fichier. En fait, prenons un exemple simple avec un fichier contenant "123456". Le pointeur de fichier peut être représenté comme un curseur que vous placeriez à une certaine position. Si vous écrivez à partir de cette position, vous remplacerez les caractères qui suivent. Le pointeur de fichier commence à la position 0, cette position correspond à un curseur placé juste avant le premier caractère du fichier. Voici comment on peut représenter ça :

Si votre pointeur de fichier est en position 0, tout ce que vous écrirez remplacera les caractères 1, 2, 3 etc ... alors que si vous aviez placé votre curseur en position 6, vous auriez écrit après et votre fichier aurait donc été "prolongé".

Maintenant nous allons mettre en pratique tous ces exemples. Si nous souhaitons ouvrir le fichier "test.txt" en lecture, on regarde d'abord le mode que nous allons utiliser. Ici, il s'agit du mode r, voici donc la syntaxe à utiliser pour ouvrir le fichier :

```
<?php     $ressource_fichier
=
fopen
(
'test.txt'
,
'r'
);
//Ouvre le fichier en lecture seule
?>
```

Notez le nom de la variable que j'ai utilisée, ici on ne récupère pas le contenu du fichier mais une ressource qui va permettre de contrôler ce fichier. On va donc ensuite pouvoir dire à PHP "utilise cette ressource et récupère le contenu du fichier". Pour récupérer le contenu du fichier, il y

a plusieurs manières :

-Récupérer caractère par caractère le contenu du fichier.
-Récupérer ligne par ligne le contenu du fichier.
-Récupérer tout le contenu du fichier d'une traite (nous avons déjà vu une méthode avec la fonction file_get_contents()).

Lire un fichier caractère par caractère
Nous allons commencer par la méthode consistant à récupérer un (ou plusieurs caractères) du fichier. La fonction permettant de retourner un seul caractère s'appelle fgetc() et prend en paramètre la fameuse ressource de fichier dont je vous ai parlé tout à l'heure.

```
<?php     $ressource_fichier
=
 fopen
(
'test.txt'
,

'r'
);

//Ouvre le fichier en lecture seule, on supposera qu'il existe sous peine d'avoir une erreur
    if
(
$ressource_fichier
)

//Si $ressource_fichier ne vaut pas FALSE on peut continuer

{
        $carac1
=
 fgetc
(
$ressource_fichier
);

//Place le chiffre 1 dans la variable $carac1
        $carac2
=
 fgetc
(
$ressource_fichier
);

//Place le chiffre 2 dans la variable $carac2
fclose
(
$ressource_fichier
);

}
?>
```

Vu que notre fichier contient "123456" si vous faites des fgetc() successivement, vous allez pouvoir récupérer le "1" puis le "2", etc ...

A chaque fois que vous utilisez la fonction fgetc(), PHP incrémente le pointeur de fichier d'une position, voilà pourquoi vous ne lisez pas le même caractère à chaque fois (ce qui ne serait pas forcément ce qu'on souhaite).

Vous imaginez bien que vous ne savez pas forcément le nombre de caractères d'un fichier, et que cette méthode n'est valable pour des fichiers contenant un nombre très restreint de caractères. On peut déjà se dire que l'on pourrait utiliser une boucle pour récupérer tous les caractères du fichier, mais il y a un problème. En effet, quoi mettre dans la condition de continuité de la boucle ? On doit dire à la boucle "arrête-toi dès que tu atteints la fin du fichier" et nous ne connaissons pour le moment aucune condition à placer pour lui dire cela. Heureusement, PHP vient à notre secours avec la fonction feof qui renvoie TRUE si le pointeur se trouve à la fin du fichier (et donc qu'il n'y a plus aucun caractère à lire). Nous utilisons également la fonction fclose() qui permet de fermer le fichier et est indispensable.

Voici donc ce que ça donne :

```
<?php     $ressource_fichier
=
 fopen
(
'test.txt'
,

'r'
);

//Ouvre le fichier en lecture seule, on supposera qu'il existe sous peine d'avoir une erreur
    if
(
$ressource_fichier
)

//Si $ressource_fichier ne vaut pas FALSE on peut continuer

{
        $contenu_fichier
=

''
;
        while
(!
feof
(
$ressource_fichier
))

//Tant que l'on est pas à la fin du fichier

{
            $contenu_fichier
.=
 fgetc
(
```

```
$ressource_fichier
);

//Récupère le caractère en cours et l'ajoute au contenu de la variable $contenu_fichier

}
        fclose
(
$ressource_fichier
);
        echo $contenu_fichier
;

//affiche le contenu du fichier

}
?>
```

Un petit rappel : $variable .= 'valeur'; est équivalent à $variable = $variable . 'valeur'; sauf que PHP interprète la première solution plus rapidement, vous y gagnez en clarté et en performances générales.

Concernant la solution de lire le fichier caractère par caractère, nous venons de le voir à titre purement pédagogique. On évitera cette solution pour des scripts couramment utilisés. Il vaut mieux lire le fichier d'une traite ou ligne par ligne, vous minimiserez ainsi les appels aux fonctions qui sont gourmands en ressources système.

Lire un fichier ligne par ligne

Deuxième méthode, cette fois-ci plus adaptée à de gros fichiers, la lecture ligne par ligne. On utilise la fonction fgets() qui permet de lire une ligne d'une seule traite. Voici ce que ça donne, je pense que vous comprendrez car cela ne change presque pas par rapport à l'exemple précédent :

```
<?php     $ressource_fichier
=
 fopen
(
'test.txt'
,

'r'
);
    if
(
$ressource_fichier
)

//Si $ressource_fichier ne vaut pas FALSE on peut continuer

{
        $contenu_fichier
=

''
;
        while
(!
```

```
feof
(
$ressource_fichier
))

//Tant que l'on est pas à la fin du fichier

{
        $contenu_fichier
.=
 fgets
(
$ressource_fichier
);

//Récupère la ligne en cours et l'ajoute au contenu de la variable $contenu_fichier

}
    fclose
(
$ressource_fichier
);
    echo $contenu_fichier
;

//affiche le contenu du fichier

}
?>
```

Lire un fichier sous forme de tableau
Vous avez déjà vu ce que sont les tableaux. Et bien PHP va vous permettre de récupérer le contenu d'un fichier dans un tableau, chaque ligne du tableau correspondra à une ligne du fichier. Ceci est très pratique quand vous avez besoin d'une ligne en particulier dans votre fichier (dont vous connaissez le numéro) ou encore pour faire des opérations de tri alphabétiques, numériques ou que sais-je encore. En tout cas, sachez que c'est également très utilisé ;)

La fonction que nous allons utiliser s'appelle file(). Comme son nom ne l'indique pas, elle va vous retourner un tableau contenant tout votre fichier. Voici comment on l'utilise :

```
<?php  $tableau
=
file
(
'test.txt'
);
//Place le contenu du fichier dans un tableau, on a supposé ici que le fichier existe sous peine d'avoir une erreur.
?>
```

Et voilà, c'est aussi simple que ça. Maintenant, il faut se souvenir d'une des fonctions que l'on peut utiliser pour parcourir un tableau :p

Bon allez je vous le dit, on peut prendre par exemple la fonction foreach(). Pour placer le contenu du fichier dans une variable sous forme de chaîne et afficher le fichier (comme nous l'avons fait pour les exemples précédents) on procèdera de cette façon :

```
<?php     $tableau
=
file
(
'test.txt'
);
//Place le contenu du fichier dans un tableau, on suppose que le fichier existe sous peine d'avoir une erreur
if(
is_array
(
$tableau
))
//Si la variable $tableau est bien un tableau, on peut continuer
    {
$contenu_fichier
=
''
;         foreach(
$tableau
AS
$ligne
)         {
$contenu_fichier
.=
$ligne
;         }         echo
$contenu_fichier
;
//Affiche le contenu du fichier, notez que l'on a pas besoin d'utiliser fclose() ici
}
?>
```

Écrire dans un fichier

Nous allons voir une seule méthode pour écrire dans un fichier, il s'agit d'utiliser la fonction fputs() disponible sur toutes les versions de PHP. Il faut d'abord ouvrir le fichier à l'aide de la fonction fopen() et un mode d'écriture approprié, ensuite on utilise fputs() pour écrire les données, et ensuite on ferme le fichier. Voici un exemple pour écrire les données "7654321" dans le fichier "test.txt" que nous avons utilisé tout au long de cette page :

```
<?php     $ressource_fichier
=
 fopen
(
'test.txt'
,

'w'
);
    if
(
$ressource_fichier
AND
is_writable
(
```

```
'test.txt'
))

//Si $ressource_fichier ne vaut pas FALSE et que le fichier est accessible en écriture alors on peut continuer

{
        fputs
(
$ressource_fichier
,
'7654321'
);
//Si une erreur a lieu, fputs() renverra FALSE, il faudra dans ce cas utiliser l'opérateur triple égal pour le savoir
        fclose
(
$ressource_fichier
);

}
?>
```

Quelques fonctions utiles
Si vous souhaitez vérifier qu'un fichier existe, vous utiliserez pour cela la fonction file_exists() qui renverra TRUE ou FALSE si le fichier existe ou non. Elle prend en paramètre le nom du fichier. Voici un exemple :

```
<?php
if(
file_exists
(
'test.txt'
))

{

echo
'Le fichier existe'
;

}    else    {         echo
'Le fichier n\'existe pas'
;    }
?>
```

Il faut toujours avant d'ouvrir un fichier, vérifier si il existe, ce que je n'ai pas fait dans mes exemples précédents, voilà pourquoi je vous mettais à chaque fois "on supposera que le fichier existe sous peine d'avoir une erreur". En effet, si vous tentez d'effectuer une ouverture d'un fichier inexistant, PHP vous renverra une erreur, ce qui serait fâcheux en terme de programmation.

Vous aurez peut-être également besoin de savoir comment effacer un fichier. En PHP, on utilise la fonction unlink() avec comme paramètre le nom du fichier à effacer. Voici comment on l'utilise :

```
<?php    if
(
file_exists
```

```
(
'test.txt'
))

{
        unlink
(
'test.txt'
);

}
?>
```

Chapitre suivant

Les expressions régulières (regex)
Présentation
Les fonctions PCRE
Recherche basique dans une chaîne
Condition "ou"
Le début et la fin d'une chaîne
Les quantificateurs
Les classes de caractères
Les métacaractères
Les classes abrégées