

Explorateur de fichiers en PHP

Par [Hugo ETIEVANT](#) 

Date de publication : 29 mars 2003

Dernière mise à jour : 29 janvier 2013

Cet article a pour but de présenter les diverses fonctions sur les dossiers ainsi que la méthode à suivre pour la gestion des fichiers et répertoires d'une arborescence à travers un cas d'école : un explorateur de fichiers en PHP.

I - Présentation

Cet article a pour but de présenter les diverses fonctions sur les dossiers ainsi que la méthode à suivre pour la gestion des fichiers et répertoires d'une arborescence à travers un cas d'école : un explorateur de fichiers en PHP.

II - Liste des fichiers d'un répertoire

Notre premier objectif est d'obtenir la liste des fichiers d'un répertoire.

II-A - La classe dir

La classe dir contient les propriétés et méthodes permettant de parcourir un dossier pour en lister les fichiers.

Propriété	Description
Handle	Valeur du pointeur vers le dossier
path	Chemin du dossier ouvert (chaîne de caractères)

Méthode	Description
dir(\$str)	Constructeur de la classe, ouvre le dossier défini par le chemin \$str (chaîne de caractères) et retourne une instance de la classe dir.
read()	Lecture d'une entrée du dossier ouvert
close()	Fermeture du dossier

Exemple :

```
<?php
$d = dir(".");
echo "Pointeur: ".$d->handle."<br>\n";
echo "Chemin: ".$d->path."<br>\n";
while($entry = $d->read()) {
    echo $entry."<br>\n";
}
$d->close();
?>
```

```
Pointeur: Resource id #1
Chemin: .
.
..
index.php3
show.php3
test.php3
images
common
tester | télécharger | voir le code source
```

Cet exemple ouvre le dossier courant . en instanciant la classe dir dans l'objet \$d dont on affiche le pointeur (*Resource id #1*) et le chemin (.). Puis, on liste les fichiers avec la méthode read() qui retourne une chaîne de caractères contenant le nom de l'entrée ou bien false s'il n'y a plus d'entrée. A chaque appel à cette fonction, un pointeur interne à l'objet \$d se déplace d'un cran dans le dossier afin d'en parcourir itérativement toutes les entrées. Une entrée peut être un fichier ou bien un autre dossier. Ne pas oublier que tout dossier même vide contient deux autres dossiers :

le dossier courant . qui renvoie au dossier actuellement ouvert, et le dossier parent .. qui permet de remonter d'un cran dans l'arborescence du système de fichiers.

II-B - Fonctions standards sur les dossiers

On peut se passer de la pseudo-classe `dir` et n'utiliser que les fonctions standards et même utiliser en plus de la classe `dir` les fonctions standards qui n'ont pas de méthodes équivalentes dans la classe.

Fonction	Description
<code>getcwd()</code>	Retourne le nom (chaîne de caractères) du dossier en cours
<code>chdir(\$str)</code>	Change de dossier courant pour aller en <code>\$str</code> , retourne <code>true</code> en cas de succès ou <code>false</code> si échec
<code>opendir(\$str)</code>	Ouvre le dossier défini par le chemin <code>\$str</code> et retourne un pointeur vers ce dossier si succès, ou bien <code>false</code> si échec
<code>closedir(\$d)</code>	Ferme le dossier identifié par le pointeur <code>\$d</code>
<code>readdir(\$d)</code>	Lit une entrée du dossier défini par le pointeur <code>\$d</code> : retourne une chaîne de caractères contenant le nom de cette entrée
<code>rewinddir(\$d)</code>	Remet à zéro le pointeur interne de parcours du dossier identifié par le pointeur <code>\$d</code>

Exemple :

```
<?php
if ($dir = opendir(".")) {
    echo "Pointeur: ".$dir."<br>\n";
    echo "Chemin: ".getcwd()."<br>\n";
    while($file = readdir($dir)) {
        echo "$file<br>\n";
    }
    closedir($dir);
}
?>
```

```
Pointeur: Resource id #1
Chemin: d:\internet\cyberzoide\php4\file
.
..
index.php3
show.php3
test.php3
test2.php3
images
common
tester | télécharger | voir le code source
```

Ici le résultat est le même qu'avec l'exemple de la classe `dir` : on ouvre le dossier avec `opendir()`, on en affiche le pointeur `$dir` et le chemin avec `getcwd()`, puis on en liste les entrées avec `readdir()` qu'on referme enfin avec `closedir()`.

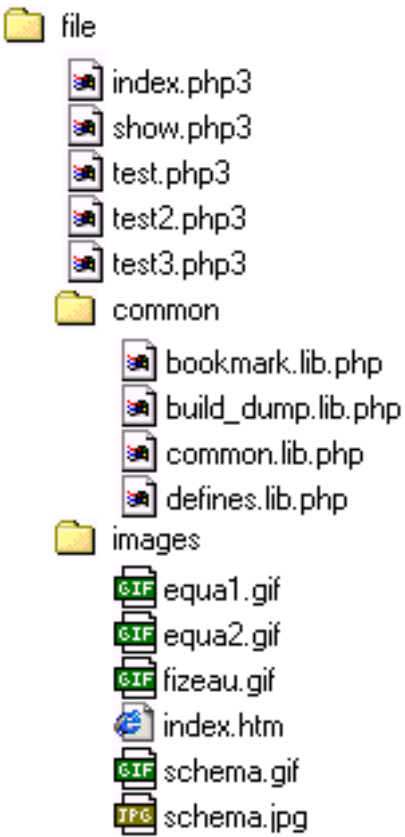
III - Liste récursive

Lister les fichiers du seul répertoire courant n'est pas très intéressant, on peut vouloir déployer l'arborescence complète d'un dossier en listant aussi les fichiers de ses sous-répertoires et ainsi de suite. Pour cela nous allons créer une

fonction `list_dir($name)` qui liste toutes les entrées du répertoire dont on passe le nom en argument. Pour chaque entrée du répertoire, on teste si elle est un dossier ou pas avec la fonction `is_dir()`. Si c'est le cas, on la liste aussi, en prenant garde à exclure les dossiers courant et parent.

Exemple :

```
<?php
function list_dir($name) {
    if ($dir = opendir($name)) {
        while ($file = readdir($dir)) {
            echo "$file<br>\n";
            if(is_dir($file) && !in_array($file, array(".", ".."))) {
                list_dir($file);
            }
        }
        closedir($dir);
    }
}
list_dir(".");
?>
```

<pre>. .. index.php3 show.php3 test.php3 test2.php3 test3.php3 common . .. bookmark.lib.php build_dump.lib.php common.lib.php defines.lib.php images . .. equa1.gif equa2.gif fizeau.gif index.htm schema.gif schema.jpg</pre>	 <p>file</p> <ul style="list-style-type: none"> index.php3 show.php3 test.php3 test2.php3 test3.php3 common <ul style="list-style-type: none"> bookmark.lib.php build_dump.lib.php common.lib.php defines.lib.php images <ul style="list-style-type: none"> equa1.gif equa2.gif fizeau.gif index.htm schema.gif schema.jpg
<p>tester télécharger voir le code source</p>	

Dans l'exemple vous avez à gauche le code source, au milieu le résultat et à droite l'arborescence complète réelle du dossier. Le script placé dans le dossier `file` parcourt l'arborescence du dossier courant et de ses fils. Il est important de tester le nom du dossier afin de ne pas parcourir le dossier courant . car cela provoquerait une boucle infinie. De plus, on interdit la remontée dans les dossiers parents .. pour préserver la confidentialité des autres dossiers du site web.

On voit que l'affichage actuel est élémentaire et l'arborescence ne saute pas yeux. On va y remédier par l'affichage de retraits à gauche. Et on en profite pour rajouter un argument `$level` à la fonction `list_dir($name, $level)`. Ce nouvel argument signale la profondeur du dossier en cours dans l'arborescence.

Exemple :

```
<?php
function list_dir($name, $level=0) {
    if ($dir = opendir($name)) {
        while($file = readdir($dir)) {
            for($i=1; $i<=(4*$level); $i++) {
                echo "&nbsp;   ";
            }
            echo "$file<br>\n";
            if(is_dir($file) && !in_array($file, array(".", ".."))) {
                list_dir($file, $level+1);
            }
        }
        closedir($dir);
    }
}
list_dir(".");
?>
```

```
.
..
index.php3
show.php3
test.php3
test2.php3
test3.php3
images
.
..
equa1.gif
equa2.gif
fizeau.gif
index.htm
schema.gif
schema.jpg
common
.
..
bookmark.lib.php
build_dump.lib.php
common.lib.php
defines.lib.php
arbo.gif
test4.php3
tester | télécharger | voir le code source
```

Dans ce nouvel exemple, on rajoute un retrait à gauche dont la longueur est proportionnel à la profondeur.

IV - Fonctions standards sur les fichiers

Nous allons maintenant aborder un exemple concret d'application de gestion d'une arborescence en programmant en PHP un explorateur de fichiers du même type que l'Explorateur Windows de Microsoft.

Avant de rentrer dans le vif du sujet, voyons quelques fonctions sur les fichiers que l'on pourra utiliser par la suite :

Fonction	Description
copy(\$source, \$dest)	Copie le fichier \$source vers \$dest et retourne true si succès ou false si echec
rename(\$source, \$dest)	Renomme le fichier \$source en \$dest et retourne true si succès ou false si echec
unlink(\$file)	Supprime le fichier \$file et retourne true si succès ou false si echec
mkdir(\$dir, \$mode)	Crée le dossier \$dir avec les droits unix \$mode (exprimés en octal) et retourne true si succès ou false si echec
rmdir(\$dir)	Supprime le dossier \$dir et retourne true si succès ou false si echec
fileatime(\$file)	Retourne la date à laquelle le fichier \$file a été accédé pour la dernière fois ou false si échec
filectime(\$file)	Retourne l'heure à laquelle le fichier \$file a été accédé pour la dernière fois ou false si échec
filemtime(\$file)	Retourne la date de dernière modification du fichier \$file ou false si échec
fileperms(\$file)	Retourne les permissions associées au fichier \$file ou false si échec
filesize(\$file)	Retourne la taille du fichier \$file (en octets) ou false si échec
filetype(\$file)	Retourne le type du fichier \$file (fifo, char, dir, block, link, file, unknown) ou false si échec
is_dir(\$file)	Retourne true si \$file est un répertoire ou false sinon
is_executable(\$file)	Retourne true si \$file est exécutable ou false sinon
is_file(\$file)	Retourne true si \$file est un fichier ou false sinon
is_link(\$file)	Retourne true si \$file est un lien symbolique ou false sinon
is_readable(\$file)	Retourne true si \$file est accessible en lecture ou false sinon
is_writable(\$file)	Retourne true si \$file est accessible en écriture ou false sinon
touch(\$file)	Change la date de dernière modification du fichier \$file en maintenant

V - Navigation

Notre explorateur doit comporter la liste des répertoires à gauche et la liste des fichiers de ce répertoire à droite. La variable \$BASE définit la racine de l'explorateur, et pour des raisons de sécurité comme de confidentialité, le visiteur ne doit pas pouvoir remonter plus loin.

La fonction list_dir(\$base, \$cur, \$level) a été modifiée de manière à ce que tous les dossiers ne soient pas automatiquement déployés. Seule la branche vers celui en cours de listage doit être déployée. L'argument \$base est le chemin complet (relatif au script) du dossier dont on doit afficher les sous-dossiers. Les noms des entrées sont exprimés en fonction de la racine \$BASE. On contrôle que chaque entrée du dossier soit un dossier autre que . et .. et on affiche une marge à gauche en fonction de la profondeur. L'argument \$cur est le dossier courant ouvert par le visiteur. Si une entrée du dossier est égale à \$cur alors inutile de placer un lien pour permettre au visiteur de l'ouvrir car il est déjà ouvert, on va plutôt le mettre en évidence en l'écrivant en **gras**. Pour développer la branche depuis la

racine \$BASE jusqu'au dossier en cours \$cur, on ne rappellera récursivement la fonction que si le nom de l'entrée (+ "/") est compris dans celui du dossier en cours (+ "/").

La fonction `list_file($dir)` permet de lister toutes les entrées (fichiers et dossiers) du dossier en cours.

Exemple :

```
<html>
<body>

<?php
$BASE = "../..";
function list_dir($base, $cur, $level=0) {
    global $PHP_SELF, $BASE;
    if ($dir = opendir($base)) {
        while($entry = readdir($dir)) {
            /* chemin relatif à la racine */
            $file = $base."/".$entry;
            if(is_dir($file) && !in_array($entry, array(".", ".."))) {
                /* marge gauche */
                for($i=1; $i<=(4*$level); $i++) {
                    echo " ";
                }
                /* l'entrée est-elle le dossier courant */
                if($file == $cur) {
                    echo "<b>$entry</b><br />\n";
                } else {
                    echo "<a href=\"\$PHP_SELF?dir=".rawurlencode($file).\">$entry</a><br />\n";
                }
                /* l'entrée est-elle dans la branche dont le dossier courant est la feuille */
                if(ereg($file."/".$cur."/")) {
                    list_dir($file, $cur, $level+1);
                }
            }
        }
        closedir($dir);
    }
}

function list_file($cur) {
    if ($dir = opendir($cur)) {
        while($file = readdir($dir)) {
            echo "$file<br />\n";
        }
        closedir($dir);
    }
}
?>

<table border="1" cellspacing="0" cellpadding="10" bordercolor="gray">
<tr valign="top"><td>

<!-- liste des répertoires
et des sous-répertoires -->
<?php
/* lien sur la racine */
if(!$dir) {
    echo "<br />";
} else {
    echo "<a href=\"\$PHP_SELF\"></a><br />";
}
list_dir($BASE, rawurldecode($dir), 1);
?>

</td><td>

<!-- liste des fichiers -->
<?php
/* répertoire initial à lister */
if(!$dir) {
```

```
$dir = $BASE;
}
list_file(rawurldecode($dir));
?>

</td></tr>
</table>

</body>
</html>
```

<ul style="list-style-type: none"> ./ . images . surfu . pourquoi . opinions . info . html . fizeau . triptique . unix . phpMyAdmin . php4 . phpMyAdmin-2.2.5 . CVS . images . CVS . lang . libraries . scripts . avatars . latex 	<ul style="list-style-type: none"> . .. CVS arrow_ltr.gif arrow_rtl.gif asc_order.gif bkg.gif browse.gif desc_order.gif fulltext.png item_ltr.gif item_rtl.gif minus.gif partialtext.png plus.gif spacer.gif
tester télécharger voir le code source	

VI - Graphismes

On a donc la trame de l'explorateur. La prochaine étape est purement graphique, il s'agit d'habiller l'application pour qu'elle ressemble d'avantage au vrai Explorateur Windows. Pour cela on va reprendre les icônes Windows suivantes :



Exemple :

```
/* listage des dossiers */
if($file==$cur) {
    echo"<img src=\"dir-open.gif\"/>&nbsp;$entry<br/>\n";
} else {
    echo"<img src=\"dir-close.gif\"/>&nbsp;$entry<br/>\n";
    echo"<a href=\"\$PHP_SELF?dir=".rawurlencode($file)."\>$entry</a><br/>\n";
}

/* listage des entrées du dossier courant */
if(is_dir($cur."/". $file)) {
    echo"<img src=\"dir-close.gif\"/>&nbsp;$file<br/>\n";
} else {
    echo"<img src=\"file-none.gif\"/>&nbsp;$file<br/>\n";
}



































/* lien sur la racine */
if(!$dir) {
```






```

echo"<img src=\"dir-open.gif\" />&nbsp;  <br />";
}else{
echo"<img src=\"dir-close.gif\" />&nbsp;  <a href=\"$PHP_SELF\"></a><br/>";
}

```

.		.	
/		.	
.		..	
images		mtools.php3	
.		unix.cpt	
surfu		scripts.php3	
.		texte.php3	
proquo		images	
.		compress.php3	
pourquoi		unix.css	
.		index.php3	
opinions		files.php3	
.		smtp.php3	
info		droits.php3	
.		sys.php3	
html		ftp.php3	
.		shell.php3	
fizeau		menu.php3	
.		scripts	
triptique			
.			
unix			
.			
images			
.			
scripts			
.			
phpMyAdmin			
.			
php4			
.			

 phpMyAdmin-2.2.5 .	
 avatars .	
 latex	
tester télécharger voir le code source	

VII - Tri des fichiers

Jusqu'à maintenant l'ordre des entrées dans les dossiers était assez obscure, ni sur le nom, ni sur le type... Quel critère d'ordre le serveur ou PHP utilise-t-il ? Il semblerait que se soit la date de création. Il serait très intéressant de pouvoir trier les fichiers sur le nom, la date, la taille, etc.

VII-A - Tri sur le nom

Comme la fonction `readdir()` ne permet pas d'imposer un critère de tri, il faut d'abord extraire tous les noms de fichier et les stocker temporairement dans un tableau et trier ce tableau avant d'afficher les listes de dossiers et de fichiers.

Dans la fonction `list_dir()`, on crée un simple tableau qui contient les entrées `$entry` lues par `readdir()`. Ainsi le `while()` se contente de remplir le tableau `$tab` des entrées qui sont des répertoires autres que `.` et `...`. Puis, on tri le tableau avec `sort()`. Et enfin, on reprend les traitements que l'on faisait dans le `while()` pour les appliquer à chacune des entrées du tableau.

Dans la fonction `list_file()`, on crée deux tableaux : un pour les dossiers et un autre pour les fichiers. Ainsi le `while()` se contente de remplir le tableau `$tab_dir` des entrées qui sont des répertoires, et le tableau `$tab_file` des entrées qui n'en sont pas. Puis, on tri les tableaux. Et enfin, on affiche le contenu des deux tableaux, l'un après l'autre car on doit afficher les dossiers avant les fichiers.

Exemple :

```


















<?php
/* racine */
$BASE = "../..";
/* liste des dossiers */
function list_dir($base, $cur, $level=0) {
    global $PHP_SELF, $BASE;
    if ($dir = opendir($base)) {
        $tab = array();
        while($entry = readdir($dir)) {
            if(is_dir($base."/".$entry) && !in_array($entry, array(".", ".."))) {
                $tab[] = $entry;
            }
        }
        sort($tab);
        foreach($tab as $entry) {
            /* chemin relatif à la racine */
            $file = $base."/".$entry;
            /* marge gauche */
            for($i=1; $i<=(4*$level); $i++) {
                echo " ";
            }
            /* l'entrée est-elle le dossier courant */
            if($file == $cur) {
                echo "<img src=\"dir-open.gif\" />&nbsp;$entry<br />\n";
            } else {



















```

```

        echo "<img src=\"dir-close.gif
\" />&nbsp; <a href=\"\$PHP_SELF?dir=\".rawurlencode(\$file).\">\$entry</a><br />\n";
    }
    /* l'entrée est-elle dans la branche dont le dossier courant est la feuille */
    if(ereg(\$file."/.", \$cur."/")) {
        list_dir(\$file, \$cur, \$level+1);
    }
}
closedir(\$dir);
}
}
/* liste des fichiers */
function list_file(\$cur) {
    if (\$dir = opendir(\$cur)) {
        /* tableaux */
        \$tab_dir = array();
        \$tab_file = array();
        /* extraction */
        while(\$file = readdir(\$dir)) {
            if(is_dir(\$cur."/".\$file)) {
                \$tab_dir[] = \$file;
            } else {
                \$tab_file[] = \$file;
            }
        }
        /* tri */
        sort(\$tab_dir);
        sort(\$tab_file);
        /* affichage */
        foreach(\$tab_dir as \$elem) {
            echo "<img src=\"dir-close.gif\" />&nbsp; <\". \$elem.\"<br />\n";
        }
        foreach(\$tab_file as \$elem) {
            echo "<img src=\"file-none.gif\" />&nbsp; <\". \$elem.\"<br />\n";
        }
        closedir(\$dir);
    }
}
?>

```

.		.	
/		..	
.			
avatars		images	
.		scripts	
fizeau		compress.php3	
.		droits.php3	
html		files.php3	
.		ftp.php3	
images		index.php3	
.			
info			
.			
latex			
.			

opinions		menu.php3	
.		mtools.php3	
php4		scripts.php3	
.		shell.php3	
phpMyAdmin		smtp.php3	
.		sys.php3	
phpMyAdmin-2.2.5		texte.php3	
.		unix.cpt	
pourquoi		unix.css	
.			
surfu			
.			
triptique			
.			
unix			
.			
images			
.			
scripts			
tester télécharger voir le code source			

VII-B - Autres tris

Tout comme le permet le véritable Explorateur Windows, nous allons permettre au visiteur de trier les listes de fichiers selon différents critères :

- le nom
- la date et heure de dernière modification extraite par filemtime()
- la taille extraite par filesize()
- le type (au sens Unix : fifo, char, dir, block, link, file, unknown) extrait par filetype()
- l'extention (synonyme du type au sens Windows) extraite par un traitement du nom
- la date de dernier accès extraite par fileatime()
- les droits d'accès extraits par fileperms()

et selon les ordres :

- croissant
- décroissant

Ainsi le tableau associatif contenant toutes les infos sur une entrée d'un dossier est généré par la fonction décrite ci-dessous. Cette fonction `addScheme()` est utilisé tant par `list_dir()` pour le listage des dossiers que par `list_file()` pour le listage des fichiers.

```
/* infos à extraire */
function addScheme($entry,$base,$type) {
    $stab['name'] = $entry;
    $stab['type'] = filetype($base."/".$entry);
    $stab['date'] = filemtime($base."/".$entry);
    $stab['size'] = filesize($base."/".$entry);
    $stab['perms'] = fileperms($base."/".$entry);
    $stab['access'] = fileatime($base."/".$entry);
    $t = explode(".", $entry);
    $stab['ext'] = $t[count($t)-1];
    return $stab;
}
```

Dans `list_dir()` seule l'information sur le nom de l'entrée sera exploitée, et seul un tri sur le nom sera effectué car la liste de gauche sur les dossiers de la racine doit être triée alphabétiquement en toute circonstance.

```
/* liste des dossiers */
function list_dir($base, $cur, $level=0) {
    global $PHP_SELF, $BASE, $order, $asc;
    if ($dir = opendir($base)) {
        $stab = array();
        while($entry = readdir($dir)) {
            if(is_dir($base."/".$entry) && !in_array($entry, array(".", ".."))) {
                $stab[] = addScheme($entry, $base, 'dir');
            }
        }
        /* tri */
        usort($stab,"cmp_name");
        foreach($stab as $elem) {
            $entry = $elem['name'];
            ...
        }
        closedir($dir);
    }
}
```

La fonction de tri `usort()` fait appel à la fonction de comparaison sur le nom `cmp_name()`.

```
function cmp_name($a,$b) {
    global $asc;
    if ($a['name'] == $b['name']) return 0;
    if($asc == 'a') {
        return ($a['name'] < $b['name']) ? -1 : 1;
    } else {
        return ($a['name'] > $b['name']) ? -1 : 1;
    }
}
```

Toutes les autres fonctions de comparaison, reposent sur le même modèle sauf qu'en cas d'égalité, elle font appel à la fonction de comparaison sur le nom.

```
function cmp_size($a,$b) {
    global $asc;
    if ($a['size'] == $b['size']) return cmp_name($a,$b);
    if($asc == 'a') {
        return ($a['size'] < $b['size']) ? -1 : 1;
    } else {
        return ($a['size'] > $b['size']) ? -1 : 1;
    }
}
```

Un titre de colonne a la forme suivante : on teste s'il est critère de tri, si c'est le cas, on test la valeur d'ordre (ascendant ou descendant) pour afficher le symbole \wedge ou \vee . Un titre de colonne porte un lien hypertexte avec les paramètres suivants : \$dir (dossier en cours, que l'on prend soin d'encoder avec rawurlencode()), \$order (critère de tri, qui peut être l'un de la liste : 'name', 'size', 'date', 'access', 'type', 'ext', 'perms'), \$asc (ordre de tri : 'a' croissant ou 'b' décroissant) et \$order0 (qui est la précédente valeur de \$order).

On liste d'abord les sous-dossiers du dossier en cours \$cur, puis les autres entrées de ce dossier (que l'on considère comme des fichiers, bien qu'il puisse il avoir aussi des liens symboliques, des flux...). On affichera les dates et heures au format francophone grâce à `date()` (car l'information extraite par les fonctions standards est le timestamp Unix), on formatera la taille avec `formatSize()` afin que sa lecture soit aisée, on affichera le type de l'entrée que l'on converti en français grâce à `assocType()`, on affiche aussi l'extention que l'on peut judicieusement remplacé par le nom de l'application associée avec `assocExt()`.

- 14 -

```

    }
    foreach($tab_file as $elem) {
        echo "<tr><td><img src=\"file-none.gif\" />&nbsp;\".$elem['name'].\"</td><td>&nbsp;\"</td><td><img src=\"file-none.gif\" />&nbsp;\".$elem['size'].\"</td><td>&nbsp;\"</td><td><img src=\"file-none.gif\" />&nbsp;\".$elem['date'].\"</td><td>&nbsp;\"</td><td><img src=\"file-none.gif\" />&nbsp;\".$elem['type'].\"</td><td>&nbsp;\"</td><td><img src=\"file-none.gif\" />&nbsp;\".$elem['ext'].\"</td><td>&nbsp;\"</td><td><img src=\"file-none.gif\" />&nbsp;\".$elem['perms'].\"</td><td>&nbsp;\"</td><td><img src=\"file-none.gif\" />&nbsp;\".$elem['access'].\"</td></tr>\n";
    }
    echo "</table>";
    closedir($dir);
}
}

```

La fonction `formatSize()` a pour but de former la taille du fichier de manière la rendre la plus simple et la plus compréhensible possible. Pour cela va afficher l'unité la plus adaptée : l'octet, le kilo octet, le giga octet ou le téra octet. Et on ne va conserver que les deux premières décimales.

```

/* formatage de la taille */
function formatSize($s) {
    /* unités */
    $u = array('octets', 'Ko', 'Mo', 'Go', 'To');
    /* compteur de passages dans la boucle */
    $i = 0;
    /* nombre à afficher */
    $m = 0;
    /* division par 1024 */
    while($s >= 1) {
        $m = $s;
        $s /= 1024;
        $i++;
    }
    if(!$i) $i=1;
    $d = explode(".", $m);
    /* s'il y a des décimales */
    if($d[0] != $m) {
        $m = number_format($m, 2, ",", " ");
    }
    return $m." ".$u[$i-1];
}

```

La fonction `assocType()` permet de donner une description francophone du type d'entité via un tableau de correspondance.

```

/* formatage du type */
function assocType($type) {
    /* tableau de conversion */
    $t = array(
        'fifo' => "file",
        'char' => "fichier spécial en mode caractère",
        'dir' => "dossier",
        'block' => "fichier spécial en mode bloc",
        'link' => "lien symbolique",
        'file' => "fichier",
        'unknown' => "inconnu"
    );
    return $t[$type];
}

```

La fonction `assocExt()` permet de donner une description francophone à l'extention d'un fichier, généralement le nom de l'application associée. Actuellement un tableau associatif, il pourra avantageusement être remplacé par un fichier csv ou une table d'une base de données. Une couleur différente pourra même être associée à chaque des extentions afin de mieux distinguer les fichiers entre eux.

```

/* description de l'extention */
function assocExt($ext) {

```

```
$e = array(
    '' => "inconnu",
    'doc' => "Microsoft Word",
    'xls' => "Microsoft Excel",
    'ppt' => "Microsoft Power Point",
    'pdf' => "Adobe Acrobat",
    'zip' => "Archive WinZip",
    'txt' => "Document texte",
    'gif' => "Image GIF",
    'jpg' => "Image JPEG",
    'png' => "Image PNG",
    'php' => "Script PHP",
    'php3' => "Script PHP",
    'htm' => "Page web",
    'html' => "Page web",
    'css' => "Feuille de style",
    'js' => "JavaScript"
);
if(in_array($ext, array_keys($e))) {
    return $e[$ext];
} else {
    return $e[''];
}
}
```

Exemple :

```
<html>
<head>
<style type="text/css">
* {font-size: 10pt;}
</style>
</head>
<body>

<?php

/* racine */
$BASE = "../..";

/* infos à extraire */
function addScheme($entry,$base,$type) {
    $tab['name'] = $entry;
    $tab['type'] = filetype($base."/".$entry);
    $tab['date'] = filemtime($base."/".$entry);
    $tab['size'] = filesize($base."/".$entry);
    $tab['perms'] = fileperms($base."/".$entry);
    $tab['access'] = fileatime($base."/".$entry);
    $t = explode(".", $entry);
    $tab['ext'] = $t[count($t)-1];
    return $tab;
}

/* liste des dossiers */
function list_dir($base, $cur, $level=0) {
    global $PHP_SELF, $BASE, $order, $asc;
    if ($dir = opendir($base)) {
        $tab = array();
        while($entry = readdir($dir)) {
            if(is_dir($base."/".$entry) && !in_array($entry, array(".", ".."))) {
                $tab[] = addScheme($entry, $base, 'dir');
            }
        }
        /* tri */
        usort($tab, "cmp_name");
        foreach($tab as $elem) {
            $entry = $elem['name'];
            /* chemin relatif à la racine */
            $file = $base."/".$entry;
            /* marge gauche */
            for($i=1; $i<=(4*$level); $i++) {
```



```

/* liste des fichiers */
function list_file($cur) {
    global $PHP_SELF, $order, $asc, $order0;
    if ($dir = opendir($cur)) {
        /* tableaux */
        $tab_dir = array();
        $tab_file = array();
        /* extraction */
        while($file = readdir($dir)) {
            if(is_dir($cur."/".$file)) {
                if(!in_array($file, array(".", ".."))) {
                    $tab_dir[] = addScheme($file, $cur, 'dir');
                }
            } else {
                $tab_file[] = addScheme($file, $cur, 'file');
            }
        }
        /* tri */
        usort($tab_dir, "cmp_".$order);
        usort($tab_file, "cmp_".$order);
        /* affichage */
        echo "<table cellspacing=\"0\" cellpadding=\"0\" border=\"0\">";
        echo "<tr style=\"font-size:8pt;font-family:arial;\">";
        <th>(((($order=='name'))?((($asc=='a'))?'/\\"
        <th>(((($order=='size'))?((($asc=='a'))?'/\\"
        <th>(((($order=='date'))?((($asc=='a'))?'/\\"
        <th>(((($order=='type'))?((($asc=='a'))?'/\\"
        <th>(((($order=='ext'))?((($asc=='a'))?'/\\"
        <th>(((($order=='perms'))?((($asc=='a'))?'/\\"
        <th>(((($order=='access'))?((($asc=='a'))?'/\\"
        foreach($tab_dir as $elem) {
            echo "<tr><td><img src=\"dir-close.gif\" />&nbsp;". $elem['name']. "</td><td>&nbsp;";
            <td align=\"right\">".formatSize($elem['size']). "</td><td>&nbsp;";
            <td>".date("d/m/Y H:i:s", $elem['date']). "</td><td>&nbsp;";
            <td>".assocType($elem['type']). "</td><td>&nbsp;";
            <td>".assocExt($elem['ext']). "</td><td>&nbsp;";
            <td>". $elem['perms']. "</td><td>&nbsp;";
            <td>".date("d/m/Y", $elem['access']). "</td></tr>\n";
        }
        foreach($tab_file as $elem) {
            echo "<tr><td><img src=\"file-none.gif\" />&nbsp;". $elem['name']. "</td><td>&nbsp;";
            <td align=\"right\">".formatSize($elem['size']). "</td><td>&nbsp;";
            <td>".date("d/m/Y H:i:s", $elem['date']). "</td><td>&nbsp;";
            <td>".assocType($elem['type']). "</td><td>&nbsp;";
            <td>".assocExt($elem['ext']). "</td><td>&nbsp;";
            <td>". $elem['perms']. "</td><td>&nbsp;";
            <td>".date("d/m/Y", $elem['access']). "</td></tr>\n";
        }
    }
}

```

```

    }
    echo "</table>";
    closedir($dir);
}
}

/* formatage de la taille */
function formatSize($s) {
    /* unités */
    $u = array('octets', 'Ko', 'Mo', 'Go', 'To');
    /* compteur de passages dans la boucle */
    $i = 0;
    /* nombre à afficher */
    $m = 0;
    /* division par 1024 */
    while($s >= 1) {
        $m = $s;
        $s /= 1024;
        $i++;
    }
    if(!$i) $i=1;
    $d = explode(".", $m);
    /* s'il y a des décimales */
    if($d[0] != $m) {
        $m = number_format($m, 2, ",", " ");
    }
    return $m." ".$u[$i-1];
}

/* formatage du type */
function assocType($type) {
    /* tableau de conversion */
    $t = array(
        'fifo' => "file",
        'char' => "fichier spécial en mode caractère",
        'dir' => "dossier",
        'block' => "fichier spécial en mode bloc",
        'link' => "lien symbolique",
        'file' => "fichier",
        'unknown' => "inconnu"
    );
    return $t[$type];
}

/* description de l'extention */
function assocExt($ext) {
    $e = array(
        '' => "inconnu",
        'doc' => "Microsoft Word",
        'xls' => "Microsoft Excel",
        'ppt' => "Microsoft Power Point",
        'pdf' => "Adobe Acrobat",
        'zip' => "Archive WinZip",
        'txt' => "Document texte",
        'gif' => "Image GIF",
        'jpg' => "Image JPEG",
        'png' => "Image PNG",
        'php' => "Script PHP",
        'php3' => "Script PHP",
        'htm' => "Page web",
        'html' => "Page web",
        'css' => "Feuille de style",
        'js' => "JavaScript"
    );
    if(in_array($ext, array_keys($e))) {
        return $e[$ext];
    } else {
        return $e[''];
    }
}

function cmp_name($a, $b) {

```

```

global $asc;
if ($a['name'] == $b['name']) return 0;
if($asc == 'a') {
    return ($a['name'] < $b['name']) ? -1 : 1;
} else {
    return ($a['name'] > $b['name']) ? -1 : 1;
}
}
function cmp_size($a,$b) {
    global $asc;
    if ($a['size'] == $b['size']) return cmp_name($a,$b);
    if($asc == 'a') {
        return ($a['size'] < $b['size']) ? -1 : 1;
    } else {
        return ($a['size'] > $b['size']) ? -1 : 1;
    }
}
function cmp_date($a,$b) {
    global $asc;
    if ($a['date'] == $b['date']) return cmp_name($a,$b);
    if($asc == 'a') {
        return ($a['date'] < $b['date']) ? -1 : 1;
    } else {
        return ($a['date'] > $b['date']) ? -1 : 1;
    }
}
function cmp_access($a,$b) {
    global $asc;
    if ($a['access'] == $b['access']) return cmp_name($a,$b);
    if($asc == 'a') {
        return ($a['access'] < $b['access']) ? -1 : 1;
    } else {
        return ($a['access'] > $b['access']) ? -1 : 1;
    }
}
function cmp_perms($a,$b) {
    global $asc;
    if ($a['perms'] == $b['perms']) return cmp_name($a,$b);
    if($asc == 'a') {
        return ($a['perms'] < $b['perms']) ? -1 : 1;
    } else {
        return ($a['perms'] > $b['perms']) ? -1 : 1;
    }
}
function cmp_type($a,$b) {
    global $asc;
    if ($a['type'] == $b['type']) return cmp_name($a,$b);
    if($asc == 'a') {
        return ($a['type'] < $b['type']) ? -1 : 1;
    } else {
        return ($a['type'] > $b['type']) ? -1 : 1;
    }
}
function cmp_ext($a,$b) {
    global $asc;
    if ($a['ext'] == $b['ext']) return cmp_name($a,$b);
    if($asc == 'a') {
        return ($a['ext'] < $b['ext']) ? -1 : 1;
    } else {
        return ($a['ext'] > $b['ext']) ? -1 : 1;
    }
}
}
?>

<table border="1" cellspacing="0" cellpadding="10" bordercolor="gray">
<tr valign="top"><td>

<!-- liste des répertoires
et des sous-répertoires -->
<?php
if(!in_array($order, array('name','date','size','perms','ext','access','type')) {

```

```

$order = 'name';
}
if (($order == $order0) && ($asc != 'b')) {
    $asc = 'b';
} else {
    $asc = 'a';
}
/* lien sur la racine */
if (!$dir) {
    echo "<img src=\"dir-open.gif\" />&nbsp;  <br />\n";
} else {
    echo "<img src=\"dir-close.gif\" />&nbsp;  <a href=\"\$PHP_SELF\"></a><br />\n";
}
list_dir($BASE, rawurldecode($dir), 1);
?>










</td><td>



















<!-- liste des fichiers -->
<?php
/* répertoire initial à lister */
if (!$dir) {
    $dir = $BASE;
}
list_file(rawurldecode($dir));
?>

</td></tr>
</table>

</body>
</html>

```

		Nom	Taille	Dernière modification	Type	Permissions	Dernier accès
/							
avatars		images	25/03/2001 17:14:36	dossier	16895		25/03/2001
emploi		scripts	08/10/2001 16:40:34	dossier	16895		08/10/2001
fizeau		unix.cpt	3 octets 26/10/2002 19:53:50	fichier	inconnu		33206 26/10/2002
html		unix.css	607 octets 06/05/2001 17:42:38	fichier			Feuille de style 33206 25/09/2002
images		index.php3	1,73 Ko 29/11/2002 16:55:40	fichier			Script PHP 33206 30/03/2003
info		menu.php3	2,26 Ko 02/11/2001 17:49:38	fichier			Script PHP 33206 26/10/2002
opinions		scripts.php3	3,10 Ko 29/11/2002 16:59:46	fichier			Script PHP 33206 29/11/2002
php4							

.		texte.php3 6,24 Ko 25/11/2002 08:45:40 fichier Script PHP 33206 25/11/2002
phpMyAdmin		
.		travail.txt 8,37 Ko 30/08/2002 13:00:56 fichier Document texte 33206 01/11/2002
phpMyAdmin-2.2.5		
.		compress.php3 10,62 Ko 29/11/2002 16:58:48 fichier Script PHP 33206 29/11/2002
pourquoi		
.		ftp.php3 10,77 Ko 29/11/2002 16:57:22 fichier Script PHP 33206 29/11/2002
proquo		
.		smtp.php3 10,99 Ko 29/11/2002 16:56:58 fichier Script PHP 33206 29/11/2002
surfu		
.		mttools.php3 11,40 Ko 29/11/2002 16:57:46 fichier Script PHP 33206 01/12/2002
triptique		
.		sys.php3 15,89 Ko 29/11/2002 16:56:02 fichier Script PHP 33206 29/11/2002
unix		
.		shell.php3 25,87 Ko 29/11/2002 16:59:12 fichier Script PHP 33206 30/03/2003
images		
.		files.php3 26,53 Ko 29/11/2002 16:56:30 fichier Script PHP 33206 29/11/2002
scripts		
		droits.php3 26,79 Ko 29/11/2002 16:58:22 fichier Script PHP 33206 29/03/2003
		tester télécharger voir le code source

VIII - Propriétés

Dans l'Explorateur Windows, un clic droit permet d'accéder aux propriétés d'un fichier ou d'un dossier. En guise de clic droit, on rajoute un lien hypertexte sur le nom des entrées du répertoire courant afin d'ouvrir le script properties.php3 dans une popup. Ce script prend pour paramètres : le type \$type de l'entrée 'dir' (dossier) ou 'file' (fichier) et le nom complet de l'entrée \$entry :

```
echo "<tr><td><a href=\"javascript:void(0);\"
onClick=\"window.open('properties.php3?type=dir&entry='. rawurlencode($cur.'/\".$elem['name']). '\", 'Proprietes',
<img src=\"dir-close.gif\" border=\"0\" />&nbsp;\".$elem['name']). \"</a></td><td>&nbsp;\"</td><td>&nbsp;\"</td>
..."
```

Le script properties.php3 extrait les informations utiles sur l'entrée choisie. Dans le cas d'un dossier, il détermine sa taille globale ainsi que le nombre de fichiers et de sous-dossiers qu'il contient via la fonction récursive getSize(). L'inclusion d'un fichier de fonctions config.php3 permet l'accès aux fonctions communes avec le script de l'Explorateur.

Exemple :

```
<html>
<head>
```

```

<title>Propriétés</title>
</head>
<body>
<?php
require_once("config.php3");

/* extraction taille totale d'un dossier,
   et calcul du nombre de fichiers et de
   dossiers contenus */
function getSize($base) {
    global $nfile, $ndir;
    $size = 0;
    /* ouverture */
    if($dir = opendir($base)) {
        /* listage */
        while($entry = readdir($dir)) {
            /* protection contre boucle infini */
            if(!in_array($entry, array(".", ".."))) {
                /* cas dossier, récursion */
                if(is_dir($base."/".$entry)) {
                    $size += getSize($base."/".$entry);
                    $ndir++;
                } /* cas fichier */
                else {
                    $size += filesize($base."/".$entry);
                    $nfile++;
                }
            }
        }
        /* fermeture */
        closedir($dir);
    }
    return $size;
}

/* dossier */
function printDir() {
    global $entry, $nfile, $ndir;

    /* extraction infos */
    $nfile = 0;
    $ndir = 0;
    $entry = rawurldecode($entry);
    $n = explode("/", $entry);
    $name = $n[count($n)-1];
    $type = assocType filetype($entry);
    $date = date("d/m/Y H:i:s", filemtime($entry));
    $size = formatSize(getSize($entry));
    $perms = fileperms($entry);

    /* affichage */
    echo "<table width=\"100%\" height=\"100%\"
    border=\"1\" bordercolor=\"gray\"
    cellspacing=\"0\" cellpadding=\"5\">
    <tr><td align=\"center\" valign=\"middle\"><table>
    <tr><td><img src=\"ico-dossier.gif\" alt=\"Dossier\" /></td><td>$name</td></tr>
    <tr><td>Type :&nbsp;</td><td>$type</td></tr>
    <tr><td>Emplacement :&nbsp;</td><td>$entry</td></tr>
    <tr><td>Taille :&nbsp;</td><td>$size</td></tr>
    <tr><td>Contenu :&nbsp;</td><td>$nfile fichiers, $ndir dossiers</td></tr>
    <tr><td>Dernière modification :&nbsp;</td><td>$date</td></tr>
    <tr><td>Attributs :&nbsp;</td><td>$perms</td></tr>
    </table></td></tr>
    </table>";
}

/* fichier */
function printFile() {
    global $entry;



    /* extraction infos */
    $entry = rawurldecode($entry);

```

```
$n = explode("/", $entry);
$name = $n[count($n)-1];
$type = assocType filetype($entry);
$date = date("d/m/Y H:i:s", filemtime($entry));
$size = formatSize(filesize($entry));
$perms = fileperms($entry);
$access = date("d/m/Y", fileatime($entry));
$t = explode(".", $entry);
$ext = assocExt($t[count($t)-1]);

/* affichage */
echo "<table width=\"100%\" height=\"100%\"
border=\"1\" bordercolor=\"gray\"
cellspacing=\"0\" cellpadding=\"5\">
<tr><td align=\"center\" valign=\"middle\"><table>
<tr><td><img src=\"ico-none.gif\" alt=\"Fichier\" /></td><td>$name</td></tr>
<tr><td>Type :&nbsp;  </td><td>$type</td></tr>
<tr><td>Emplacement :&nbsp;  </td><td>$entry</td></tr>
<tr><td>Taille :&nbsp;  </td><td>$size</td></tr>
<tr><td>Extension :&nbsp;  </td><td>$ext</td></tr>
<tr><td>Dernière modification :&nbsp;  </td><td>$date</td></tr>
<tr><td>Dernier accès :&nbsp;  </td><td>$access</td></tr>
<tr><td>Attributs :&nbsp;  </td><td>$perms</td></tr>
</table></td></tr>
</table>";
}

switch($type) {
case 'dir' : printDir(); break;
case 'file' : printFile(); break;
}
?>
</body>
</html>
```

	Php4
Type :	dossier
Emplacement :	.././php4
Taille :	15,36 Mo
Contenu :	269 fichiers, 24 dossiers
Dernière modification :	10/06/2001 13:53:48
Attributs :	16895
	ezpdf.pdf
Type :	fichier
Emplacement :	.././php4/ezpdf/ezpdf.pdf
Taille :	485,96 Ko
Extension :	Adobe Acrobat
Dernière modification :	09/03/2003 16:04:26
Dernier accès :	09/03/2003
Attributs :	33206
Fichier maître : http://cyberzoide.developpez.com/php4/file/test9.php3 télécharger voir le code source	
Fichier d'édition des propriétés : télécharger voir le code source	
Bibliothèque de fonctions : télécharger voir le code source	