# Aide mémoire PHP

Michel Meynard

9 janvier 2006

## 1 Introduction

- PHP : langage de script incorporé dans du HTML, fonctionnant coté serveur, soit comme CGI, soit comme un module (+rapide).
- Langage interprété, non pré-compil, donc référence avant impossible (sauf PHP4)
- Expressions régulières riches, conversions automatiques
- Déclarations optionnelles et non typées
- Fin de ligne (;) obligatoires
- Fonctions natives pour accéder aux SGBD (MySQL, Oracle, PostgresSQL, ODBC, ...). Connexions persistantes (**A FERMER**) ;
- Variables préfixées par le symbole : $
- Syntaxe fonctionnelle : `fopen("/home/toto/file.txt", "r") ;`
- PHP est un logiciel libre
- Possibilité de télécharger des fichiers depuis le client
- Extensions (librairies dynamiques) : gd (graphique), analyseur XML, ...
- Gestion des id de sessions par cookie ou par URL longue (PHP4)

### 1.1 Exemple : fichier `hello.php`

```
<html><head><title>Example</title></head><body>
<?php echo "Hello World !"; ?>
</body></html>
```

## 2 Documentation

### 2.1 Sites

**téléchargement** `http ://www.php.net/`

**Doc. en Français** `http://dev.nexen.net/docs/php/annotee/manuel_toc.php`

**FAQ** `http ://www.php.net/FAQ.php`

**aide sur PHP** `http ://www.phphelp.com/`

**cours en Français** `http://www.linux-france.org/article/devl/php3/tut/php3.tut.html`

## 3 Développement

### 3.1 Généralités

Commentaires par : `/* ... */` ou `// ...` ou `# ...`. Edition du source `toto.php` puis interprétation par le serveur http. Exécution sur la ligne de commande du serveur : `php -q test.php`. Débogage manuel, par exemple :

```
$DEBUG=1; // débogage activé dans le script
function debug($msg){if ($GLOBALS['DEBUG'])
  echo $msg, <br>"\n";}
debug("ligne ". __LINE__ ." x = ".$x); // appel
```

### 3.2 Programmation modulaire

Possibilité d'inclusion de fichier par `require("fic.php") ;` ou `include("fic.php") ;`. Préférer `include` qui est conditionné par l'exécution de la ligne le contenant. En PHP4, `include_once()` et `require_once()` permetent d'éviter les redéfinitions. Une extension PHP est une bibliothèque dynamique (`php3_xx.dll` ou `php3_xx.so`) pouvant être chargée à la volée ou incorporée à la compilation de php. Par exemple : `dl('php3_gd.so') or die('impossible de charger gd !') ;` // gd est une extension de gestion des images gif, png.

## 4 Types, constantes et variables

Les types sont : boolean, integer, float (flottant), string, array, object, resource, NULL. Coercition de type implicite ou explicite par `(type) exp`. Pas de déclaration. Les variables sont définies dans un contexte d'exécution, fonction (locales) ou script (globales). Les variables globales (script) ne sont accessibles dans une fonction que si on les déclare globales dans la fonction : `global $i ;`. Les variables statiques sont des variables locales à une fonction mais qui sont initialisées une seule fois lors du premier appel à la fonction. Elles doivent être déclarées : `static $compte=0 ;`. Les variables "superglobales" sont des variables globales qui sont accessibles dans les fonctions sans qu'on ait besoin de les déclarer `global`.

### 4.1 Variables prédéfinies

Les variables prédéfinies dépendent de la configuration de php : pour les connaître, le plus simple est de faire appel à `phpinfo()`.

**$PHP_SELF** nom du fichier php en cours d'exécution ; également accessible via `$_SERVER["PHP_SELF"]`.

Un certain nombre de variables prédéfinies sont des tableaux superglobaux :

**$GLOBALS** tableau associatif des variables globales ;

**$_ENV** variables d'environnement du serveur ;

**$_GET** variables obtenues par la méthode HTTP GET ;

**$_POST** variables obtenues par la méthode HTTP POST ;

**$_COOKIE** variables de cookie envoyées par le client au serveur ;

**$_SERVER** tableau associatif des variables du serveur ; par exemple, `$_SERVER["REQUEST_METHOD"]` indique la méthode utilisée (GET ou POST OU ...) ;

**$_FILES** variables utilisées lors du chargement (upload) de fichiers ;

**$_REQUEST** variables de tous les tableaux précédents avec la priorité définie par la directive de configuration `variables_order` qui vaut habituellement `EGPCS` ;

### 4.2 Variables de formulaire

Les variables de `formulaire HTML` sont accessibles via les tableaux superglobaux `$_GET` ou `$_POST`. Par exemple :

```
<form name='F' method='get' action='
<?php echo $_SERVER['PHP_SELF']; ?>'>
<input type='text' name='nom'>
<select multiple name="biere[]">
 <option value="blonde">blonde</option>
 <option value="brune">brune</option>
</select>
<input type='submit' name='bouton' value='ok'/>
</form>
<?php print_r($_GET['biere']); ?>
```

Après saisie et validation, l'url suivante est requise :

```
essai.php?nom=toto&biere[]=blonde&
biere[]=brune&bouton=ok
```

L'affichage suivant aura lieu :

```
Array ( [0] => blonde [1] => brune )
```

### 4.3 Littéraux

**chaîne** `'$a et ‘ls‘'` sans interprètation

**chaîne** `"$a et ‘ls‘"` avec interprètation

**échappement** `\$,\\,\",\t,\n,\r` sans interprètation

**numérique** 123, 123.456, 2e10, `\777,\xFF`

**contexte booléen** ” ou 0 ou "0" représentent FALSE ;

**conversion contextuelle** automatique chaîne-numérique

### 4.4 Constantes

Macro définie grâce à la fonction `define("K",10);echo K;`
Constantes prédéfinies :

**__FILE__** nom du fichier exécuté

**__LINE__** numéro de ligne courante

**TRUE** valeur TRUE (différent de 0)

**FALSE** valeur FALSE : 0 ou "" ou "0" ou tab vide ou objet vide

### 4.5 Tableaux

Que des tableaux **associatifs** avec une clé entière (0..n-1) par défaut. Les tableaux ont un pointeur courant automatiquement incrémenté lors d'une affectation.

**définition** `$tab = array (1, 2, 6); $tvide = array();`

**définition** `$asso = array ("toto" => 1, "titi" => 8);`

**affectation multiple** `list($i,$j)=($j,$i)` ou `list($i,$j)=$tab`

**ajout** `$tvide[]="case 0";$tvide[]="case 1";`

**raz** `reset($tab);`

**fin de tableau** `end($tab)` : va pointer sur le dernier élément

**parcours** `array each($tab)` : retourne une asso à 4 éléments ou faux si fin de tableau : $r[0] = $r["key"] = clé courante ; $r[1] = $r["value"] = valeur courante. Exemple : `reset($tab); while (list ($c, $v) = each ($tab)){echo "$c => $v";}`

**accès** `$tab[6]=$asso["toto"];`

**valeur courante** `mixed current($tab)` : faux si vide ou en fin ou valeur FALSE.

**valeur et déplacement** `mixed next($tab); mixed prev($tab);` retourne la valeur courant puis délacement, faux si place vide ou fin ou début de tableau

**taille** `int count($tab);` ou sizeof()

**clé courante** `mixed key($tab)`

**existence** `bool in_array ("toto",$tab)`

**2 dimensions** `$et=array(array(0,0),array(0,1));`

**autres fons** des quantités ! intersection (`array_intersect`), fusion (`array_merge`), différence (`array_diff`) ...

## 5 Expressions

Quasiment tout est une expression en PHP.

### 5.1 Opérateurs

Quasiment utilisés que pour les **nombres** sauf la concaténation de chaîne (.) et l'affectation (chaînes et tableaux).

**$$s** valeur de la variable dont le nom est dans s

++, - - inc et déc : `++$x;$y- -;`

**$i==$j ? 1 : 2** expression conditionnelle

**arithmét.** `+,-,*,/,%`

**affectation** `=,.=,+=,-=,*=,/=,%=`

**binaires** `&,|,~,<<,>>`

**logiques** `&&,||,!`

**logique peu prioritaire** `and,or`

**comparaison** `<,<=,==,!=,>=,>`

**commande syst.** `‘ls‘`

**concaténation** `$s . "toto"`

L'affect. de chaîne longue est possible (PHP4) :

```
$s=<<<FIN
bla bla..
etc
FIN; // en début de ligne et avec un ‘‘;’’
```

## 6 Structures de contrôle

**alternative** `if (expr) {instons1} else {instons2}`

**choix** `if (e1) {ins1} elseif (e2) {ins2} else {ins3}`

**choix** PHP4 `switch ($i) {case e0 : instons0 ; break ; case e1 : instons1 ; break ; default : instons ; }`

**répétitives** `while (e) {instons}`, `do { instons } while (e)`, `for (exp-init ; exp-cond ; exp-itér) { instons }`, ruptures possibles : `break ;`, `continue ;`

**parcours de tableau** `foreach($tab as $case) { instons utilisant $case}`

**parcours de tableau** `foreach($tab as $cle => $val) { instons utilisant $cle ou $val }`

# 7 Fonctions

Un nombre impressionnant de fonctions prédéfinies ! Passage des paramètres scalaires ou tableaux par valeur (PHP4 par référence si précédé de & dans l'appel et la déclaration). Les variables globales ne sont pas accessibles directement. Il faut utiliser la notation : `$GLOBALS['glob']`. `$GLOBALS` est l'asso. des variables globales. On peut aussi déclarer les variables globales dans la fon par `global $i,$j ;` Les arguments peuvent avoir une valeur par défaut (optionnels) et être en nombre variable. Les valeurs de retour peuvent être scalaire ou tableaux. On peut supprimer l'affichage des erreurs produites par l'appel à une fonction f par : `@f($i)`

## 7.1 Fonctions utilisateur

**définition** `function f($arg0="toto", $arg1){instons ; return array(1, 4) ;}`

**pointeur** `$pf='f';list($i,$j)=$pf(1,2) ;`

# 8 Fonctions prédéfinies

## 8.1 Divers

**echo** exp1, exp2, ... ; affiche les expressions ;

**print(exp1)** affiche une expression ;

**print_r($var)** affiche la variable scalaire ou tableau ou objet récursivement ;

**void exit()** termine le script

**$l=system("ls") ;** exécute une commande

**void die ("message ")** affiche le message puis termine

**void eval("instons PHP")** il faut échapper les caractères spéciaux : `eval('$' . f($i) . '= "toto";');` Ne pas oublier le " ;"

## 8.2 Entrées/Sorties : système de fichier

De nombreuses fonctions de gestion du système de fichier.

**echo string, string ... ;** affiche plusieurs arguments **chaînes**

**print(string) ;** affiche un arg. (**classé** dans les fons chaînes)

**$desc=fopen("fic.txt","r+") ;** ouverture en lecture et écriture. Modes (r, w (création ou raz), w+ , a (append), a+).

**$d=fopen("c : \\data\\info.txt","r") ;** Windows

**$d=fopen("ftp ://user :password@example.com/", "w") ;** ouverture de session ftp

**fclose($desc)** fermeture

**bool feof($desc)** test la fin

**$ligne=fgets($desc, 4096)** lecture de la ligne (maxi 4096 octets)

**$ligne=readline("un entier SVP") ;** ne fonctionne pas sur le Web !

**$car=fgetc($desc) ;** lecture d'un car ;

**fputs($desc,"chaine") ;** écriture d'une chaîne ;

## 8.3 Types et variables

**boolean isset($i)** vrai si i est défini

**unset($i)** supprime la définition

**string gettype($i)** retourne le type (string si non défini)

**void settype($i,"integer")** coercition

**int is_int($i)** is_double(), is_string(), is_array()

**var_dump(exp1, exp2, ...)** affiche récursivement la valeur et le type des expressions ;

## 8.4 Tableaux

**int array_push($pile,$elem)** empile à la fin et ret la taille

**mixed array_pop($pile)** dépile le sommet

**int array_unshift($fifo,$prem)** ajoute au début

**mixed array_shift($fifo)** retire le premier

**array array_values($asso)** resp. `array_keys`

**void shuffle($tab)** mélange les valeurs

## 8.5 Tris de tableau

**void sort($tab)** tri croissant (décroissant avec rsort) selon les valeurs

**void ksort($asso)** tri croissant selon les clés (krsort)

**void asort($asso)** tri croissant selon les valeurs (arsort)

**void usort($tab, moncmp)** tri croissant selon la fon de cmp définie par l'utilisateur

**uksort, uasort** tri utilisateur pour les asso

## 8.6 Chaînes

De très nombreuse fonctions dont voici les principales. `$s` est supposée être une chaîne.

**int strlen($s)** taille

**int strcmp($s1,$s2)** comparaison -1, 0, 1

**string substr($s, 2, 10)** sous-chaîne à partir de 2, de taille 10

**string strstr($s,$facteur)** ret tout s à partir de la 1ère occurrence de facteur

**array split(';',$s,10)** ret un tableau des 10 (maxi) premiers champs séparés par des ;

**string join(';',$tab)** ret la chaîne constituée des valeurs séparées par ;

**string trim($s)** retire les blancs de début et de fin de chaîne. Blancs : `\n,\r,\t,\v,\0,espace`

**string chop($s)** supprime les blancs de fin de chaîne

**string ltrim ($s)** supprime les blancs de début

**string strtolower($s)** met en minuscules (resp. `strtoupper`)

**string nl2br($s)** remplace les `\n` par des `<br>`

### 8.7 Expressions régulières

**correspondance** `int ereg('^[^.]*\.(.*)$',$s,$tabr)` met dans tabr la partie à droite d'un point dans s à partir de l'indice 1. Retourne TRUE si trouvé au moins une.

**remplacement** `string ereg_replace("([a-z]+)", "[\\1]",$s);` encadre les mots minuscules de s

## 9 Classes

Collection (tableau) de variables et de fonctions; héritage simple possible par extends; tout est public; l'affectation se fait par recopie; pas de surcharge (utiliser un paramètre tableau).

```
class Point{var $x,$y;
  function Point($px=0,$py=0){ //constr
    $this->x=$px;$this->y=$py;
  }
  function getxy(){
    return array($this->x, $this->y);
  }
}
$p1=&new Point(); // (0,0)
$p2=&new Point(3,4);
list($a,$b)=$p2->getxy();
```

## 10 Bases de données

Les fonctions PHP d'accès aux bd sont nombreuses. Il existe des fonctions spécialisées pour des sgbds tels que MySQL ou Oracle ou PostgreSQL, mais aussi des fonctions ODBC qui sont plus indépendantes du sgbd cible. Enfin, il existe une librairie `PHPLib` (`http ://phplib.netuse.de/`) contenant une classe `DB_Sql` générique pour les BD (comme `DBI` en Perl).

### 10.1 MySQL

Fonctions spécifiques à MySQL :

**int mysql_connect('hostname','username','password')** `or die('Connexion au serveur impossible !');` ret un descripteur de connexion `$co`. La connexion se termine en fin de script.

**int mysql_pconnect('hostname','username','password')** connexion **persistante** : si une connexion persist. existe déjà avec le même nom et passwd, la réutilise; ne ferme pas la connexion en fin de script. `$co`

**int mysql_select_db('mabase',$co)** `or die('Base de données inaccessible !');` ret vrai si sélectionné (`use`)

**int mysql_query('select|insert| ...',$co)** `or die('Requête impossible');` retourne faux si impossible; retourne un entier résultat (`$res`) si requête select; retour inutile si requête action

**int mysql_num_fields($res)** ret nb de colonnes du résultat d'un select

**string mysql_field_name($res,$i)** ret le nom de la colonne d'indice i

**string mysql_field_type($res,$i)** ret le type. (`mysql_field_len`, `flags` existent également)

**int mysql_num_rows($res)** ret nb de lignes

**array mysql_fetch_assoc($res)** ret une ligne sous forme d'asso. indicé par les noms de colonnes

**array mysql_fetch_row($res)** ret une ligne sous forme de tableau indicé de 0 à mysql_num_fields - 1

**int mysql_affected_rows($co)** ret le nombre de lignes affectées par la dernière requête `DELETE, INSERT, REPLACE, UPDATE`

**int mysql_db_query ('base2','select ...',$co)** sélectionne une bd et pose une requête

**string mysql_error($co)** ret la chaîne indiquant l'erreur MySQL

**int mysql_close($co)** ferme la connexion (**INDISPENSABLE**)

### 10.2 Un exemple

```
$co=mysql_connect('sql.free.fr','monnom',
  'monpasswd') or die('Connexion au serveur
  impossible !');
mysql_select_db('mabase',$co) or die("Base de
  données inaccessible !");
$r=mysql_query('select nom, prenom from matable',
  $co) or die('Requête impossible');
$nc=mysql_num_fields($r); $nl=mysql_num_rows($r);
while ($ligne=mysql_fetch_assoc($r)){
  echo "NOM : ",$ligne['nom'],"; PRENOM : ",
  $ligne['prenom'],"<BR>\n";
  }
mysql_close($co);
```

## 11 phpMyAdmin

phpMyAdmin est un outil d'administration d'une BD MySQL écrit en PHP. Cet outil est accessible via le web et permet donc d'administrer à distance une BD. On peut : créer, supprimer, modifier (alter) des tables, interroger, ajouter, supprimer, modifier des lignes, importer des fichiers textes contenant les données ... Le site de référence où l'on peut tester en direct est : `http://phpwizard.net/projects/phpMyAdmin/index.html`. Site de Free : `http://phpmyadmin.free.fr/phpMyAdmin/`.

## 12 Réseau

**socket** `$sock=fsockopen($hostname,$port,$timeout)` ouvre une socket; il ne reste plus qu'à lire dedans ... et la fermer !

## 13 Images (gd)

Nécessite d'avoir compilé avec le module GD !

```
<?
header("Content-type: image/png");
$larg=100;$haut=200;
$img=imagecreate($larg,$haut);
$red=imagecolorallocate($img,255,0,0);
$vert=imagecolorallocate($img,0,255,0);
imagefilledrectangle($img,0,0,$larg-1,$haut/2,$red);
imageline($img,0,0,$larg-1,$haut-1,$vert);
```

```
imagepng($img);//stdout
imagedestroy($im);
?>
```

# 14 Caractères spéciaux

- Certains caractères (anti-slash, guillemet et apostrophe) saisis dans des champ de formulaires (`<input name="champ">`) sont automatiquement **échappés** par un anti-slash par PHP (magic_quotes). La variable Php correspondante `global $champ;` doit donc être traitée par `$champ=stripslashes($champ)` pour supprimer **tous** les anti-slash générés.
- Lors de l'affichage d'une chaîne contenant des caractères html spéciaux (`&"<'>`) dans un champ, certains caractères et ceux qui suivent ne sont pas affichés et sont perdus (cela ne sera pas posté). Pour afficher proprement une chaîne contenant ces caractères, il faut au préalable la traiter avec `htmlspecialchars($champ,ENT_QUOTES))`.
- Par conséquent, pour un champ interactif, on écrira : `echo '<input name="champ" value="'.htmlspecialchars(stripslashes($champ),ENT_QUOTES)).'">';`
- Lors de l'envoi d'une requête à un SGBD, il faut parfois échapper les caractères spéciaux tels que : `',\,",NULL`. La fonction `addslashes($chaine)` effectue ce travail.
- De même en JavaScript, la fonction `addslashes()` permetra d'échapper du code.

# 15 Administration PHP

Pour visualiser la configuration PHP, il suffit d'appeler la fonction `phpinfo()` pour voir :
- la version de php ;
- la localisation du fichier de configuration de php : `php.ini` ;
- les librairies incluses et leur configuration (mysql, gd, ...).

Le fichier `php.ini` définit notamment des variables fondamentales :

**register_globals=On** si **on** alors les variables d'Environnement, Get, Post, Cookie, Serveur sont globales ; si **off** alors les variables sont accessibles via `$_POST[], ...` (Sécurité +) ;

**variables_order="EGPCS"** ordre de résolution des conflits de noms de variables Env, GET, POST, COOKIE, Serveur ;

**magic_quotes_gpc=On** permet d'anti-slasher les caractères anti-slash, guillemet et apostrophe dans les variables GPC.

# 16 Authentification HTTP

La procédure d'authentification HTTP est associée à un **nom de domaine** (realm ou AuthName) et à un répertoire. Elle peut être déclenchée :
- soit par Apache, indépendamment de PHP ;
- soit en utilisant PHP.

## 16.1 Authentification Apache

L'authentification est valable pour toute une **arborescence**. Un fichier `.htaccess` spécifie les règles d'authentification valables pour ce répertoire et tous ses **descendants**.

```
AuthType Basic
AuthUserFile "/nfs1/.../AuthApache/.htpasswd"
AuthName "Le Domaine Privé"
<Limit GET POST>
 require valid-user
</Limit>
```

Le fichier `.htpasswd` contient la liste des utilisateurs et leurs mots de passe cryptés. Il est obtenu grâce à l'utilitaire `htpasswd` fourni par Apache. Par exemple : `htpasswd -c .htpasswd un` ; crée un fichier avec l'utilisateur `un`.
Lors de tout accès à un fichier descendant de `AuthApache`, Apache va envoyer un en-tête au navigateur client qui va afficher une fenêtre d'authentification. Par la suite, l'utilisateur reste authentifié pour "Le Domaine Privé" jusqu'à la fin du navigateur ou si une nouvelle authentification PHP est lancée.

## 16.2 Authentification PHP

PHP doit être un module d'Apache. On utilise alors la fonction `header` pour demander une authentification ("WWW-authenticate") au client, générant ainsi l'apparition d'une fenêtre de demande d'utilisateur et de mot de passe. Une fois que les champs ont été remplis, l'URL sera de nouveau appelée, avec les variables `$_SERVER['PHP_AUTH_USER']`, `$_SERVER['PHP_AUTH_PW']` et `$_SERVER['PHP_AUTH_TYPE']` contenant respectivement le nom d'utilisateur, le mot de passe et le type d'authentification. Actuellement, seule l'authentification de type "Basic" est supportée. Si l'authentification est réalisée via Apache (`.htaccess`), la variable `$_SERVER['REMOTE_USER']` est égale à `$_SERVER['PHP_AUTH_USER']`.
Exemple d'authentification HTTP par PHP :

```
<?php
if (!isset($_SERVER['PHP_AUTH_USER']) || !verifi
erAuth())
  header("WWW-Authenticate: Basic realm=\"Le Dom
aine Privé\"");
  header("HTTP/1.0 401 Unauthorized");
  echo "Texte à envoyer si le client annule\n";
  exit;
  } else {
  echo "Bonjour ",$_SERVER['PHP_AUTH_USER'];
  // suite de la page privée
  }
?>
```

La fonction booléenne `verifierAuth()` utilisera tout moyen nécessaire à la vérification du nom et du mot de passe (Base de données, ...). Remarquons que les variables `$_SERVER['PHP_AUTH_...']` sont utilisables même si l'authentification n'a pas été effectuée par le module PHP.
**Les variables d'authentification PHP sont valables pour tous les fichiers descendants du répertoire**. Par contre, tout fichier html ou php ne testant pas l'authentification est accessible, contrairement à l'authentification par .htaccess (Apache) qui sécurise tout le répertoire.

## 16.3 Désauthentification PHP

Le navigateur écrase le cache d'authentification client d'un domaine quand il reçoit une nouvelle demande d'authentification. Cela permet de déconnecter un utilisateur, pour

le forcer à entrer un nouveau nom et son mot de passe. Si l'utilisateur annule l'authentification, il est alors désauthentifié ! Penser à recharger la page. Certains programmeurs changent dynamiquement le nom de domaine pour donner un délai d'expiration, ou alors, fournissent un bouton de réauthentification.

### 16.4 Autres authentifications

L'authentification HTTP est de moins utilisée au profit d'un formulaire d'authentification HTML et de la sauvegarde des informations d'authentification dans un cookie ou une session. En effet, le fonctionnement des différents navigateurs varie en ce qui concerne le cache d'authentification.

## 17 Téléchargement

- du site web vers le client : download ou déchargement ;
- du client vers le site web : upload ou chargement ;

### 17.1 Déchargement

Réaliser un lien référençant le fichier à décharger, par exemple : <a href="Public/toto.ps">Cliquer ici</a>. Si le type du fichier est affichable par le navigateur, il sera affiché (donc déchargé), sinon il sera proposé à l'utilisateur soit de sauver le fichier, soit de lancer l'application associée. Remarquons que tout lien peut être enregistré en utilisant le bouton droit de la souris sur le lien.

### 17.2 Chargement

Réaliser un formulaire de chargement envoyant une requête POST. L'action effectuée (script) après soumission doit vérifier que le fichier chargé est du bon type et de la bonne taille puis l'afficher depuis la zone temporaire (tmp/) ou il est chargé. Le fichier temporaire sera **automatiquement effacé** de la zone à la fin du script, s'il n'a pas été déplacé ou renommé. Par exemple, le formulaire sera :

```
<FORM ENCTYPE="multipart/form-data" ACTION="traite
ment.php" METHOD="POST">
  <INPUT TYPE="hidden" NAME="MAX_FILE_SIZE" value=
"1000">
  Envoyez ce fichier : <INPUT NAME="fichier" TYPE=
"file" SIZE=15>
  <INPUT TYPE="submit" VALUE="Envoyer le fichier">
</FORM>
```

Dans le script `traitement.php` on a accès à différentes variables (différent selon les versions de PHP) :

- `$_FILES['fichier']['name']` le nom du fichier original chez le client ;
- `$_FILES['fichier']['type']` le type mime du fichier "image/gif, text/plain, ..." ;
- `$_FILES['fichier']['size']` la taile du fichier chargé ;
- `$_FILES['fichier']['tmp_name']` le nom du fichier temporaire sur le serveur ;
- `$_FILES['fichier']['error']` le code d'erreur (PHP 4.2.0).

Remarquons que tous les navigateurs ne vérifient pas le `MAX_FILE_SIZE`.

## 18 Session

HTTP n'étant pas un protocole orienté connexion, chaque nouvelle requête semble être la première. Depuis PHP4, Les **sessions** PHP permettent de conserver de l'information **du côté serveur** sur la suite de requêtes émises par le même client (navigateur). Les variables enregistrées dans le tableau `$_SESSION` doivent être de type `string`. Pour les variables de type tableau, elles doivent être sérialisées. La communication des identifiants de sessions est réalisée par cookie ou par URL de manière transparente pour le programmeur. Par défaut, la durée de vie du cookie est égale à la durée de vie du navigateur.

**bool session_start()** démarre une session ; doit être réalisé en tout début de script avant tout en-tête !

**bool session_register("x", "y", ... )** démarre une session si pas encore fait et y enregistre les variables $x et $y ;

**string session_id()** retourne l'identifiant de session

**$_SESSION["z" ]**="contenu"; ajoute une variable de session ;

**echo $_SESSION["x" ]**; affiche le contenu d'une variable de session ;

**bool session_is_registered("mavar")** teste si $mavar a été sauvé dans la session ;

**bool session_unregister("x")** supprime la variable de session x ;

**bool session_destroy()** supprime toutes les données de la session ; le cookie de session sera supprimé dès la prochaine page.

**string serialize(mixed)** retourne une chaîne composée des différents champs de l'objet ou du tableau ;

**mixed unserialize(string)** retourne la valeur de la variable d'origine ;

## 19 Cookies

Les cookies sont stockées du côté client par le navigateur et contiennent des variables. Quand celui-ci envoie une requête à une URI d'un domaine et d'un chemin dont il possède un cookie, toutes les variables du cookie sont envoyées au serveur. Un cookie a une durée de vie (expire) par défaut égale à la durée de vie de la session.

**int setcookie("x","Hello !",int expire, string path, string domain, int secure)** doit être exécutée avant toute chose et permet de positionner la variable x avec la valeur "Hello !" ; attention le cookie ne sera renvoyé au serveur qu'à la prochaine requête ;

**$_COOKIE["x" ]** accès à une variable de cookie ;

**setcookie("x","",time()-1000)** efface la variable x du cookie ;

**setcookie("y",$value,time()+3600) ;** expire dans une heure ;

## 20 Redirection

La fonction php `header` doit être exécutée avant toute chose et permet de nombreuses choses dont :

**la redirection** vers une autre URL :

```
header("Location: http://www.php.net/");
/* Redirige le client vers le site PHP */
exit(); /* rien après */
```

**l'erreur** header("Status : 404 Not Found") ;

**l'expiration** header("Expires : Mon, 26 Jul 2003 05 :00 :00 GMT") ;

**la date de maj** header("Last-Modified : " . gmdate("D, d M Y H :i :s") . " GMT") ;

**la non mise en cache** header("Cache-Control : no-cache, must-revalidate") ;

**le type mime** header("Content-type : application/pdf") ;

**le nom du fichier** header("Content-Disposition : attachment ; filename=toto.pdf") ;