

Structure générale d'un script PHP

Les instructions PHP sont réunies dans un bloc entouré de balises spécifiques du fichier HTML. Comme toutes les balises HTML celle ci commence par le symbole "<" et se termine par le symbole ">". Le bloc contenant du PHP est caractérisé par en début et fin par le symbole "?". Tout ce qui est à l'extérieur de la balise PHP est transmis tel quel au navigateur ; seul les instructions PHP sont interprétées par le moteur PHP.

Remarque le nouveau format XML demande que la balise d'ouverture des instructions PHP soit "php" et non plus "?" seulement.</p

Le langage PHP n'est pas un langage classique de programmation dans la mesure où les instructions, ne sont pas regroupées en programme ou procédure mais sont dispersées dans la page HTML, entourées de balises appropriées, le but de ces instructions étant de construire une page HTML "classique" après avoir pris en compte les demandes reçues du client.

L'instruction principale du langage consistera donc à compléter le fichier HTML, après soit des calculs soit des requêtes à une base de données, cette instruction commence par le mot clé **echo**, exemple affichage de la date dans une page HTML :

```
<html>
<head>
<title>Affichage Date</title>
</head>
<body>
Nous sommes le

    <?php
    /*
        Affichage jour mois année
    */
    echo Date("d/m/y");

?>

</body>
</html>
```

Seule la partie en italique sera interprétée par le moteur PHP. La partie entre /**/ ne sera pas prise en compte : ce sont des commentaires sensés améliorer la lisibilité du script, la seule instruction ici est l'instruction **echo** qui fait écrire dans le fichier envoyé au client ce qui se trouve derrière ce mot clé, ici le résultat d'une fonction PHP qui donne la date du jour, la partie entre guillemets, à l'intérieur des parenthèses, indique comment doit être écrite cette date.

La plupart du temps nous aurons à traiter des données pour répondre à la demande du client, il est donc nécessaire de stocker ces données élémentaires dans des zones mémoire du serveur pour leur appliquer ces traitements, comme tous les langages de programmation, PHP utilise des variables pour ce faire. D'autre part les traitements peuvent être soumis à certaines conditions ou porter sur un ensemble important de données, en cas par exemple d'interrogation de base de données, le langage PHP est un langage structuré qui fournit des structures conditionnelles et répétitives permettant de faire ces traitements.

Variables, types, opérateurs.

Un autre type d'instruction très usuel est le stockage de valeurs intermédiaires pour un traitement ; ce stockage se fait dans des zones mémoire du serveur identifiées par un nom de variable. Un nom de variable en PHP est une suite de caractères commençant par le symbole "\$". Le nom de variable doit respecter les règles suivantes (après le symbole \$) :

- Le premier caractère doit être une lettre ou le caractère souligné _ (underscore)
- Les autres caractères doivent être des chiffres des lettres ou le caractère souligné.
- Les caractères tels que + , - , * , = , < , > , ! , & ne sont pas autorisés
- Enfin PHP tient compte de la casse les variables \$x et \$X sont considérées comme deux variables distinctes.

Instruction d'affectation de valeur à une variable

Pour affecter une valeur à une variable, on utilise le signe =, la variable qui doit recevoir une valeur se trouve à gauche de ce signe, à droite se trouve une constante ou une expression faisant éventuellement intervenir des variables, cette instruction est suivie du signe ; symbole de fin d'instruction, exemples :

```
$nom = "Dupont";  
$age=102;  
$marie=True;
```

Type de données

PHP n'est pas un langage déclaratif, les variables ne peuvent pas être déclarées par l'utilisateur, cependant suivant les valeurs prises par la variable PHP va lui attribuer un type. Ce type peut éventuellement être changé par l'utilisateur, comme nous le verrons plus loin.

Les types reconnus et associés aux variables par PHP sont les suivants :

- Le type chaîne de caractères (texte)
- Le type entier (numérique)
- Le type virgule flottante (réel, numérique)
- Le type booléen (Vrai ou Faux)
- Le type tableau
- Le type objet
- Le type ressource associé à des opérations système telles que connexion à une base de données, ouverture d'un flux de fichier.

Type de données chaînes de caractères

Le type chaîne de caractères contient tout ce qui est une suite de caractères alphanumérique ou numérique (mot, phrase etc..).

Pour affecter une valeur constante sous forme de chaîne de caractère il faut la mettre entre guillemets (simples ou doubles) :

```
$auteur="Jean Paul Sartre";  
$departement ='75';
```

la variable \$departement sera considérée comme une chaîne de caractères et devra donc être convertie en numérique si on veut effectuer des opérations arithmétiques sur elle (cette conversion sera éventuellement automatique, voir plus loin).

La seule opération possible sur les chaînes de caractères est la concaténation c'est à dire le recollement de deux ou plusieurs chaînes de caractères. Le signe opératoire est le ".", par exemple :

```
$nom= "Vian";  
$prénom="Boris";  
$auteur=$prénom." ".$nom;  
echo $auteur;
```

affichera :

Boris Vian

PHP, comme le langage C, autorise une notation abrégée quand l'un des opérandes de la concaténation est aussi la variable réceptrice :

$\$a=\b ; est équivalent à $\$a=\$a.\$b$; attention à l'ordre de la concaténation!

Remarque 1: Les doubles guillemets et les simples guillemets ne se comportent pas toujours de la même façon, il faut donc être très prudent dans leur utilisation, par exemple l'instruction

```
Echo "$nom";
```

Affichera Vian

Tandis que

```
Echo '$nom';
```

Affichera \$nom

Remarque 2 : la concaténation peut s'appliquer à des variables non chaînes de caractères, PHP fait alors la transformation de type (typecasting) :

```
$num=40;
```

```
echo $num." ans";
```

Affichera 40 ans

Types de données numériques

Il existe en PHP deux types de variables numériques les nombres entiers et les nombres réels (à virgule flottante). Les nombres entiers sont ceux qui correspondent pour les autres langages de programmation aux entiers longs (généralement entre $-2\,147\,483\,648$ et $+2\,147\,483\,647$). Les nombres à virgule flottante sont composés de nombres négatifs compris entre $-1,79769313486232 \times 10^{308}$ et $-4,940654584124 \times 10^{-324}$, du nombre 0 et de nombres positifs compris entre $4,940654584124 \times 10^{-324}$ et $1,79769313486232 \times 10^{308}$, c'est à dire que l'on environ 13 chiffres significatifs.

L'affectation de données numériques à une variable se fait sans guillemet, le séparateur décimal est le point :

$\$i = 12$; définira une variable entière

$\$x = 30.245$; définira une variable réelle.

Opérations arithmétiques simples

Les opérateurs usuels +, -, * et / sont définis en PHP, avec les priorités habituelles.

Il existe en plus l'opérateur %, opérateur modulo qui donne le reste de la division entière d'un nombre par un autre :

$\$reste = 34 \% 5$; donnera 4 comme résultat

cette opération n'a de sens que pour les nombres entiers, PHP fera la troncature des nombres décimaux si nécessaire :

$\$reste = 34.6 \% 5.7$; donnera aussi 4 comme résultat.

PHP, comme pour les chaînes de caractères, autorise certaines notations abrégées particulières :

$\$a = \$a + \$b$; peut s'écrire $\$a += \b ;

$\$a = \$a - \$b$; peut s'écrire $\$a -= \b ;

$\$a = \$a * \$b$; peut s'écrire $\$a *= \b ;

$\$a = \$a / \$b$; peut s'écrire $\$a /= \b ;

$\$a = \$a \% \$b$; peut s'écrire $\$a \% = \b ;

Enfin $\$a = \$a + 1$; peut s'écrire $\$a++$; et $\$a = \$a - 1$; peut s'écrire $\$a--$; ces dernières écritures sont utilisées fréquemment dans les structures répétitives.

Type de données booléennes

Le type de données booléen correspond à une variable ne pouvant prendre que deux valeurs vrai ou faux. L'affectation d'une variable booléenne se fait soit en lui donnant l'une de ces deux valeurs :

$\$test = \text{TRUE}$; ou $\$test = \text{FALSE}$;
soit en lui affectant le résultat d'une comparaison.

Opérateurs de comparaison

Les opérateurs de comparaison sont les suivants :

| Exemple | Nom | Résultat |
|---------------|-------------------|---|
| $\$a == \b | Egal | TRUE si $\$a$ est égal à $\$b$. |
| $\$a === \b | Identique | TRUE si $\$a$ est égal à $\$b$ et s'ils sont de même type |
| $\$a != \b | Différent | TRUE si $\$a$ est différent de $\$b$. |
| $\$a <> \b | Différent | TRUE si $\$a$ est différent de $\$b$. |
| $\$a !== \b | Différent | TRUE si $\$a$ est différent de $\$b$ ou bien s'ils ne sont pas du même type. |
| $\$a < \b | Plus petit que | TRUE si $\$a$ est strictement plus petit que $\$b$. |
| $\$a > \b | Plus grand | TRUE si $\$a$ est strictement plus grand que $\$b$. |
| $\$a <= \b | Inférieur ou égal | TRUE si $\$a$ est plus petit ou égal à $\$b$. |
| $\$a >= \b | Supérieur ou égal | TRUE si $\$a$ est plus grand ou égal à $\$b$. |

Exemple pour l'opérateur $==$: si vous définissez ainsi les deux variables $\$a$ et $\$b$:

$\$a = 123$;
 $\$b = "123"$;

la comparaison $\$a == \b donnera **TRUE**, PHP faisant la conversion dans un même type; tandis que la comparaison $\$a === \b donnera **FALSE**.

Ces opérateurs de comparaison sont surtout utilisés dans les structures conditionnelles et répétitives.

Opérateurs booléens

PHP dispose des opérateurs booléens standards, dont la liste est donnée ci-dessous :

| Exemple | Nom | Résultat |
|------------------------|--------------------|---|
| $! \$a$ | NON (<i>Not</i>) | TRUE si $\$a$ est faux. |
| $\$a \&\& \b | ET (<i>And</i>) | TRUE si $\$a$ ET $\$b$ sont vrais. |
| $\$a \ \ \b | OU (<i>Or</i>) | TRUE si $\$a$ OU $\$b$ est vrai. |
| $\$a \text{ and } \b | ET (<i>And</i>) | TRUE si $\$a$ ET $\$b$ sont vrais. |
| $\$a \text{ xor } \b | XOR (<i>Xor</i>) | TRUE si soit $\$a$ soit $\$b$ est vrai, mais pas les deux en même temps. |
| $\$a \text{ or } \b | OU (<i>Or</i>) | TRUE si $\$a$ OU $\$b$ est vrai |

La raison pour laquelle il existe deux types de "ET" et de "OU" est qu'ils ont des priorités différentes, le tableau donne les opérateurs en ordre décroissant de priorité.

Comme pour les autres opérateurs, il est possible d'utiliser des notations abrégées :

$\$a \&= \b ; pour $\$a = \$a \&\& \$b$;
 $\$a \|\|= \b ; pour $\$a = \$a \|\| \$b$;
 $\$a \wedge= \b ; pour $\$a = \$a \text{ xor } \$b$;

Le type tableau

En PHP, un tableau est en fait une association ordonnée, c'est à dire une correspondance entre une clé et une valeur, par défaut les clés sont les entiers positifs (0,1,2,...). Contrairement aux langages classiques pour un même tableau :

- Le tableau n'a pas besoin d'être dimensionné, il ne peut donc y avoir de débordement.
- La clé peut indifféremment être une chaîne de caractère ou un entier positif.
- Le type des valeurs du tableau n'est pas nécessairement le même pour toutes les clés et peut être un tableau ce qui permet de créer des tableaux multidimensionnels.

Ce type permet donc d'être assimilé à plusieurs objets : vecteur matrice, liste, dictionnaire, arbre etc.

Il existe deux méthodes principales pour créer un tableau, l'affectation directe de valeurs aux éléments du tableau ou l'utilisation de la fonction `array()`.

Affectation de valeurs à un tableau

Pour faire référence à un élément d'un tableau, il faut indiquer le nom de la variable tableau suivi de l'indice entre crochets :

```
$array[0]=12.50;
$array[5]="bonjour";
$array["civilité"]="Monsieur";
$array["livraison"]["rue"]="Avenue du bois";
$array["livraison"]["numéro"]=12;
$array["livraison"]["code postal"]="94122";
```

Dans le même tableau, ici `$array`, il est possible donc d'avoir différents type de clés, chaîne de caractères ou entier, et différents type de valeurs associées à ces clés : numériques entiers ou flottants, chaînes de caractères, ou tableau ici pour la clé "livraison".

Attention : il n'est pas donné de valeur par défaut aux clés manquantes, même pour les clés nombres entiers, c'est à dire qu'ici une instruction du type :

`Echo $array[3]` provoquera l'erreur :

Notice: *Undefined offset: 3 in g:\easyphp1-7\www\essais\plante1.php on line 21.*

Il est possible d'affecter de nouvelles valeurs à un tableau sans préciser la clé, dans ce cas la clé générée par PHP est la valeur entière maximum actuelle +1, 0 si aucune clé entière positive n'existe. Sur l'exemple précédent, l'instruction :

```
$array[]=345;
```

créera une clé 6.

Donc pour créer un tableau unidimensionnel "classique", dont les indices commencent à 0, il suffit d'une suite d'instruction de type :

```
$tableau[]=..;
```

Pour tester si une valeur est affectée à une clé du tableau on utilisera la fonction **isset()**. Pour supprimer un élément d'un tableau, on utilisera la fonction **unset()**, par exemple **unset(\$array["livraison"]);** supprimera les trois valeurs du tableau.

Nous verrons dans le paragraphe sur les structures répétitives comment explorer toutes les valeurs d'un tableau.

Utilisation de la fonction `array()`

Pour créer un tableau, on peut aussi utiliser la fonction **array()** dont la syntaxe est la suivante :

```
array( [key =>] value, ...);
```

où `key` est une chaîne de caractère ou un entier et peut être omis, `value` peut être n'importe quoi. Si `key` est omis, `key` sera une suite d'entiers positifs.

Exemples :

```
$tableau=array(2,8,16); donnera $tableau[0]=2, $tableau[1]=8, $tableau[2]=16.
```

Pour créer le tableau du paragraphe précédent, on pourrait utiliser l'instruction :

```
$arr= Array ( 0 => 12.5, 5 => "bonjour", "civilité" => "Monsieur", "livraison" => Array ( "rue" => "Avenue du bois", "numéro" => 12, "code postal" => "94122" ), 345 );
```

le fait que le dernier élément n'ait pas de clé explicite provoquera, comme dans le cas de création par affectation, la clé 6.

Remarque sur les clés :

La déclaration d'une clé chaîne de caractère ne contenant que des chiffres entre guillemets n'est prise en compte comme telle que si le nombre commence par un 0. Par exemple "5" créera la clé numérique 5 et non pas la chaîne de caractères, tandis que "05" créera la clé chaîne de caractère associée.

Affichage de la structure d'un tableau

Il peut être utile d'afficher toute la structure d'un tableau, pour ce faire on utilise l'instruction `print_r`. Sur l'exemple des paragraphes précédents l'instruction :

```
print_r($array);
```

provoquera l'affichage :

```
Array ( [0] => 12.5 [5] => bonjour [civilité] => Monsieur [livraison] => Array ( [rue] => Avenue du bois [numéro] => 12 [code postal] => 94122 ) [6] => 345 )
```

Remarquez que

- 1 - les clés non numériques ne sont pas entre guillemets, mais entre crochets, alors qu'il est nécessaire d'utiliser les guillemets dans le programme PHP,
- 2 - l'absence des virgules séparatrices, nécessaires aussi dans le programme PHP.

Utilisation "non standard" d'un tableau

Nous allons terminer ce paragraphe sur les tableaux en donnant une illustration d'utilisation du type tableau en PHP pour représenter des piles. Une pile est une structure *last in-first out*, un nouvel élément est inséré au-dessus de la pile. C'est donc le dernier élément entré dans la pile qui sera supprimé en premier. Cette structure peut donc se représenter de façon récursive sous la forme : `pile=[tete, pile]`.

Pour entrer un nouvel élément `$a` dans la pile associée à la variable `$pile`, il suffira des deux instructions suivantes :

```
$pile["pile"]=$pile;
```

```
$pile["tete"]=$a;
```

la suppression d'un élément est encore plus simple :

```
$pile=$pile["pile"];
```

si ce dernier n'est pas vide, il sera donc en fait nécessaire de faire un test.

A titre d'exercice nous laissons au lecteur le cas symétrique de la queue, qui correspond à une structure *first in-first out*, définie récursivement par `queue=[premier,queue]` et celui de l'arbre binaire, définie par `arbre=[arbre_gauche, nœud, arbre_droit]`.

Les variables prédéfinies en PHP

Depuis la version 4.1.0, PHP fournit un jeu de tableaux prédéfinis, contenant les variables du serveur (si possible), les variables d'environnement et celle d'entrées. Ces nouveaux tableaux sont un peu particuliers, car ils sont automatiquement disponibles dans tous les environnements d'exécution. Pour cette raison, ils sont dits 'auto-globaux' ou bien encore 'superglobaux' (il n'y a pas de mécanisme PHP pour créer de telles variables). Nous allons

lister certaines de ces tableaux avec leurs clés les plus utiles. Nous ne parlerons ici que de la variable `$_SERVER`, laissant à d'autres paragraphes, l'utilisation des variables `$_POST`, `$_GET`, `$_COOKIE` et `$_SESSION`.

La variable `$_SERVER`

Le tableau `$_SERVER` contient de nombreux renseignements utiles sur le serveur ou la page affichée.

Prenons l'exemple de l'URL : "http://www.monsite.com/chemin/test.php ?param=valeur".

Le tableau indique les clefs utilisées dans `$_SERVER["clé"]` et le résultat correspondant.

| clé | résultat |
|-----------------------------------|---|
| <code>PHP_SELF</code> | <code>/chemin/test.php</code> |
| <code>SERVER_NAME</code> | <code>www.monsite.com</code> |
| <code>REQUEST_URI</code> | <code>/chemin/test.php ?param=valeur</code> |
| <code>QUERY_STRING</code> | <code>param=valeur</code> |
| <code>HTTP_USER_AGENT</code> | <code>Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322)</code> |
| <code>HTTP_ACCEPT_LANGUAGE</code> | <code>fr</code> |
| <code>DOCUMENT_ROOT</code> | exemple <code>/home/monsite/www</code> = la racine au niveau du système de fichier |
| <code>SCRIPT_NAME</code> | <code>/chemin/test.php</code> |
| <code>SCRIPT_FILENAME</code> | <code>/home/monsite/www/chemin/test.ph</code> = <code>\$DOCUMENT_ROOT + \$SCRIPT_NAME</code> |
| <code>SERVER_ADDR</code> | L'adresse IP du serveur |
| <code>REMOTE_ADDR</code> | L'adresse IP du visiteur |
| <code>HTTP_REFERER</code> | La page qui a pointé vers la page actuelle (la page précédente) |
| <code>REQUEST_METHOD</code> | La méthode ayant servi pour accéder à la page : GET, POST, HEAD, PUT |

Il existe d'autre clés pour ce tableau, le programme suivant vous permettra de les obtenir ainsi que les valeurs associées (voir le chapitre des structures pour la boucle sur le tableau) :

```
<?
// essai de la variable server
foreach($_SERVER as $i=>$value)
{
    echo $i." , ".$_SERVER[$i]."<br>";
}
?>
```

Les structures de programmation

Une instruction élémentaire PHP peut être sur plusieurs lignes et plusieurs instructions élémentaires peuvent être sur la même ligne. Chaque instruction se termine par un ";".

Un bloc d'instruction, c'est à dire un ensemble d'instructions élémentaires, est entouré d'accolades, l'accolade fermante n'a pas besoin d'être suivie d'un ";".

La structure de base est la structure séquentielle, c'est à dire que chacune des instructions est exécutée une fois et une seule dans l'ordre de leur écriture. Cependant dans certains cas certaines instructions n'ont à être exécutées que dans certaines circonstances (sous condition), dans ce cas on parlera de structure conditionnelle, et dans d'autres cas des instructions doivent être exécutées plusieurs fois, on parle alors de structure répétitive.

Structure conditionnelle

Il existe deux types d'instructions conditionnelles la structure alternative et la structure à choix multiples.

La structure alternative

La structure alternative consiste à exécuter certaines instructions si une condition est vérifiée et d'autres si elle ne l'est pas ; il est possible que l'une des deux branches de l'alternative soit vide.

L'instruction if

L'instruction if permet d'exécuter une ou plusieurs instruction si une condition est vérifiée, sa syntaxe est la suivante :

```
<?php
if (expression)
instruction(s)
?>
```

où expression est une expression booléenne. Si il y a plusieurs instructions à exécuter si la condition est vérifiée, ces instructions doivent être regroupées en un bloc entre accolades.

Exemple

```
<?php
if ($a>$b)
echo "$a est plus grand que b";
?>
```

ou avec plusieurs instructions :

```
<?php
if ($a>$b)
{
    echo "$a est plus grand que b";
    $sup=$a;
    $inf=$b;}
?>
```

Remarque importante : pour l'égalité les opérateurs usuels sont == ou ===, il faut faire attention à l'opérateur d'affectation que l'on peut utiliser dans une condition :

```
if($a=$b){...}
```

Est en fait équivalent à :

```
$a=$b;
```

Pour l'instruction

```
if{ $a}{....}
```

\$a est converti en booléen, dans le cas où il n'est pas booléen, sa valeur est FALSE si :

- \$a est numérique et vaut 0

- \$a est une chaîne vide
- \$a est un tableau vide

Dans tous les autres cas \$a est TRUE.

L'instruction if ... else

Si d'autres instructions sont à exécuter si la condition n'est pas vérifiée, il faudra faire suivre l'instruction if d'une instruction *else*. La syntaxe est la suivante :

```
<?php
if (expression)
{instruction1(s)}
else
{instruction2(s)}
?>
```

Dans ce cas même si instruction1 est unique elle doit être entourée d'accolades, la même règle s'applique pour instruction2.

Exemple :

```
<?php
if ($a>$b)
{
    $sup=$a;
    $inf=$b;
}
else
{
    $sup=$b;
    $inf=$a;
}
?>
```

Syntaxe particulière : dans le cas où instruction1 et instruction2 sont tous deux réduits à une seule instruction, il est possible d'utiliser la syntaxe alternative :
(condition) ? instruction1 : instruction2;

Structure à choix multiples.

L'instruction elseif

Cette instruction permet de tester des conditions disjointes et d'exécuter des instructions associées à une et une seule de ces conditions. La syntaxe est la suivante :

```
<?php
if (condition1)
{ instruction1}
elseif(condition2)
{instruction2}
elseif(condition3)
{instruction3}
....
Else
{instructionfin}
?>
```

Seules les instructions correspondant à la réalisation de la condition i sont exécutées, si aucune des conditions n'est réalisée, les instructions écrites entre accolades après le else sont exécutées.

Exemple :

```
<?php
if ($a=== $b)
{echo "$a et $b sont égaux et de même type";}
elseif($a== $b)
{echo "$a et $b sont égaux mais de types différents";}
else
{echo "$a n'est pas égal à $b";}
?>
```

L'instruction switch

Lorsque les conditions portent sur l'égalité à certaines valeurs, il est possible d'utiliser une autre syntaxe, l'instructions switch :

```
switch ($variable)
{
    case value1:
        instruction1;
        break;
    case value2:
        instruction2;
        break;
    ....
    default:
        instructionn;
}
```

Remarque : si le mot clé break est omis dans chacun des cas des qu'une condition est vérifiée toutes les instructions suivantes sont exécutées, jusqu'à la rencontre éventuelle du mot break ou la fin du switch.

Les structures répétitives

Nous présenterons ici deux structures répétitives universelles la boucle for et la boucle while, et une structure particulière aux tableaux la boucle foreach.

L'instruction for

La boucle for n'est pas indispensable, mais elle permet une programmation plus aisée que la boucle while quand à chaque exécution de la boucle, chaque itération, on passe à l'itération suivante en ajoutant une valeur fixe entière (positive ou négative) à un compteur et que l'arrêt de la boucle se fait quand le compteur dépasse une valeur donnée.

La syntaxe est la suivante :

```
<?php
for(initialisation; continuation; incrémentation)
instruction(s)
?>
```

si instructions comprend plusieurs instructions celles ci doivent être mises entre accolades.

- Initialisation est une instruction affectant une valeur initiale au compteur.
- Continuation est une expression booléenne faisant intervenir le compteur, qui autorise la continuation de la boucle tant qu'elle est vraie.
- Incrémentation est une instruction indiquant la façon dont le compteur est modifié à chaque itération.

Au démarrage l'initialisation du compteur est effectuée, puis le test de continuation est évalué, si la valeur est FALSE, les instructions ne sont pas exécutées et le programme sort de la boucle, sinon les instructions sont exécutées et le compteur est incrémenté pour l'itération suivante.

Le cas le plus courant est une boucle du type :

```

<?php
for($compteur=debut; $compteur<max; $compteur++)
instruction(s)
?>

```

qui correspond à l'ajout de 1 au compteur, à chaque itération. Pour soustraire 1 au compteur à chaque itération on utiliserait \$compteur--, en lieu et place de \$compteur++.

Si l'on voulait utiliser un autre pas pour le compteur, on écrirait :

```

<?php
for($compteur=debut; $compteur<max; $compteur+=pas)
instruction(s)
?>

```

Exemple de boucles for imbriquées : construction d'un tableau HTML à 5 lignes et 4 colonnes contenant les 20 premiers entiers.

```

<table border="1" align="center">
<?php
// initialisation de l'élément mis dans le tableau HTML
$element=0;
// $li compteur de ligne
for($li=0;$li<5;$li++)
{
// balise de début de ligne
?>
<tr>
<?php
// $col compteur de colonnes
for($col=0;$col<4;$col++)
{
// balise de début de cellule
?>
<td>
<?php
$element++;
// écriture de l'élément
echo $element;
// balise de fin de cellule
?>
</td>
<?php
// fin de la boucle sur $col
}
// balise de fin de ligne
?>
</tr>
<?php
// fin de la boucle sur $li
}
// balise de fin de tableau
?>
</table>

```

Ce qui donne le résultat suivant :

| | | | |
|----|----|----|----|
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 |
| 13 | 14 | 15 | 16 |
| 17 | 18 | 19 | 20 |

L'instruction while

La boucle *while* est une boucle plus générale. Un ensemble d'instruction est exécuté tant qu'une condition est vérifiée. Cette boucle est très utile quand on ignore combien de fois la boucle doit être exécutée ou plus précisément quand on ne pas lier l'exécution de la boucle à un compteur, par exemple lecture d'un fichier jusqu'à la rencontre de la fin de fichier, traitement du résultat d'une requête sur une base de donnée etc.

La syntaxe est la suivante :

```
<?php
while (condition)
instruction(s)
?>
```

si instructions comprend plusieurs instructions celles ci doivent être mises entre accolades. Condition représente une expression booléenne qui doit, bien évidemment, être modifiée à chaque itération. Comme pour le if, la condition peut-être une affectation suivie d'une conversion : `while($a=$b){ ... }`, il faut bien faire attention à ce que \$b soit modifié au cours des itérations.

Exemple, ce petit programme compte le nombre de jour, entre le jour courant et le premier vendredi suivant :

```
<?php
//récupère le temps du jour en seconde (à partir du temps 0 du calendrier :1 janvier 1970)
$jour=time();
// initialise à 0 le nombre de jours
$nb=0;
//date("l",$jour) donne le nom du jour
while (date("l",$jour)!="Friday")
{
// ajouter 1 au nombre de jours
$nb++;
//passer au jour suivant (60*60*24 secondes par jour)
$jour=$jour+60*60*24;
}
echo "Il reste $nb jours";
?>
```

L'instruction foreach

L'instruction *foreach* permet d'explorer tous les éléments d'un tableau dont les clés sont inconnues aussi bien en nombre qu'en valeur. Cette instruction peut prendre deux formes suivant que l'on veuille ou non utiliser les clés du tableau. Dans tous les cas l'exploration du tableau se fait dans l'ordre de la création des valeurs du tableau et non pas dans l'ordre croissant des clés.

La première forme permet d'explorer uniquement les valeurs du tableau. La syntaxe est la suivante :

```
<?php
foreach($nomdutableau as $elementdutableau)
instruction
?>
```

instruction étant une instruction simple ou un bloc d'instructions entre accolades.

\$nomdutableau est le nom de la variable associée à un tableau existant, tandis que

\$elementdutableau est une variable qui va recevoir les différentes valeurs du tableau, valeurs qui seront utilisée dans le bloc d'instruction..

Exemple :

```
<?php
$arr[2]="l'ami";
$arr[3]="Martin";
```

```
$arr[1]="Bonjour";
foreach($arr as $elt)
echo "$elt<br>";
?>
```

provoquera l'affichage :

```
l'ami
Martin
Bonjour
```

Appliqué à l'exemple du paragraphe sur les tableaux le résultat sera le suivant :

```
12.5
bonjour
Monsieur
Array
345
```

Array correspondant au cas où l'une des valeurs est un tableau (ici la valeur correspondant à la clé "livraison").

La deuxième forme de l'instruction foreach permet d'obtenir en plus la clé du tableau, la syntaxe est la suivante :

```
<?php
foreach($nomdutableau as $cle=>$elementdutableau)
instruction
?>
```

instruction étant une instruction simple ou un bloc d'instruction entre accolades.

\$nomdutableau est le nom de la variable associée à un tableau existant,

\$cle est une variable qui va recevoir les différentes valeur de la clé,

\$elementdutableau est une variable qui va recevoir les différentes valeurs du tableau, valeurs qui seront utilisée dans le bloc d'instruction..

Exemple :

```
<?php
$arr[2]="l'ami";
$arr[3]="Martin";
$arr[1]="Bonjour";
foreach($arr as $cle=>$elt)
echo "$cle - $elt<br>";
?>
```

donnera comme résultat :

```
2 - l'ami
3 - Martin
1 - Bonjour
```

Mais le problème demeurera pour l'exemple du paragraphe sur les tableaux.

Pour résoudre ce problème il faut utiliser la notion de fonction récursive, pour le lecteur averti voici une solution et l'affichage résultant (la fonction `gettype($variable)` retournant une chaîne de caractères correspondant au type de la variable):

```
<?php
//fonction récursive d'écriture d'un élément
function ecrit($a)
// on teste si le type de la variable est un tableau
{if (gettype($a)=="array")
//si oui on explore les éléments du tableau avec un retrait positif à gauche
{echo "<blockquote>";
foreach ($a as $cle=>$elt)
{echo "<li>$cle : ";
ecrit($elt);}
echo "</blockquote>";}
// sinon on affiche l'élément
else{echo "$a <br>";}
}
```

```
//exemple d'application
$arr= Array ( 0 => 12.5, 5 => "bonjour", "civilité" => "Monsieur",
"livraison" => Array ( "rue" => "Avenue du bois", "numéro" => 12, "code postal" => "94122" ), 345
);
ecrit($arr);
?>
```

le résultat étant :

- 0 : 12.5
- 5 : bonjour
- civilité : Monsieur
- livraison :
 - rue : Avenue du bois
 - numéro : 12
 - code postal : 94122
- 6 :345

Dialogue client serveur

Nous allons aborder ici différentes façons d'échanger des informations avec le client, les formulaires et les cookies.

Les formulaires

Les formulaires correspondent aux zones de saisie sur une page HTML, le contenu de ces zones peut-être passé au serveur principalement sous deux formes, soit par l'URL après un "?", c'est la méthode GET, soit dans l'en-tête de la requête http envoyée au serveur, c'est la méthode POST, c'est cette méthode que nous utiliserons.

La balise de formulaire contient le nom de la page PHP requise pour la réponse (dans la propriété action), ainsi que la méthode utilisée :

```
<form method="POST" action="traite_formu.php" name="Saisie des données" >
```

Un formulaire est composé de plusieurs éléments de saisie, et sera créé à l'aide d'un outil de construction de site web tel que Frontpage ou Dreamweaver. Chaque zone de saisie, appelée contrôle, comporte une balise qui correspond à la nature du contrôle, une propriété "name" qui sera utilisé pour la récupération des données, une propriété "value" qui correspond à l'entrée du client et quelquefois une propriété type qui spécialise le contrôle. Le fichier correspondant à la propriété action devra donc traiter ces valeur, après les avoir récupérer dans une variable superglobale \$_GET ou \$_POST suivant la méthode employée.

Les principaux contrôles et leur récupération par le serveur

Dans le cas de la méthode POST toutes les valeurs envoyées par le client sont stockées dans la variable tableau superglobale **\$_POST**, les clés de ce tableau étant les noms des contrôles du formulaire. Il est possible d'utiliser directement ce tableau, mais très souvent il est préférable de stocker la valeur du tableau dans une variable après avoir vérifier que la valeur existe avec la fonction **isset()**.

Les zones de texte

Les zones de texte correspondent à des zones où le client entre un texte d'une ligne maximum, la propriété value retournera le texte entré par l'utilisateur. La balise du contrôle est "input" type de ce contrôle est "text".

Récupération de la variable :

```
<?php
if(isset($_POST[nomtxt]))
$txt=$_POST[nomtxt];
?>
```

Les espaces de texte

Les espaces de texte permettent de saisir plusieurs lignes de texte. La balise de ce contrôle est "textarea". La récupération se fera comme pour la zone de texte.

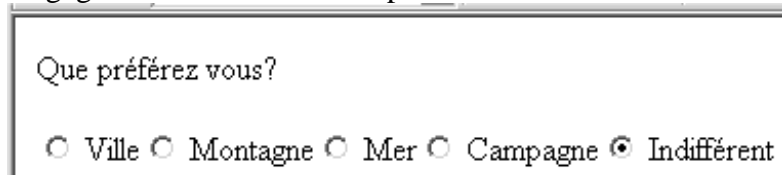
Les cases d'option

Ces contrôles, appelés en anglais radio buttons, correspondent à un choix unique parmi plusieurs possibilités, l'utilisateur cliquera sur l'un des choix. La balise de ce contrôle est aussi "input", le type de ce contrôle est "radio", tous les boutons correspondant à l'ensemble des choix portent le même nom, il est souvent commode d'affecter à la propriété "value" de chaque bouton correspond à l'intitulé du choix. Pour indiquer un choix par défaut on ajoutera checked au bouton correspondant.

Voici un script qui permet de créer un ensemble de cases d'option, avec les intitulés correspondants:

```
Que préférez vous?</p>
<?php
// tableau contenant les intitulés mis en valeurs
$lieux=array("Ville","Montagne","Mer","Campagne","Indifférent");
for($i=0;$i<5;$i++)
{
?>
<input type="radio" value=
<?php
echo $lieux[$i];
// le choix par défaut est le dernier
if ($i==4){
//ici l'accolade est nécessaire pour insérer du HTML
?>
checked
<?php
;}// pour finir le if
?>
name="choix">
<?php
//écriture de l'intitulé sur la page HTML
echo $lieux[$i];
}
//fermeture de la boucle for
?>
```

Les lignes en italique correspondent à du HTML, on voit ici qu'un script PHP peut ne pas être en un seul morceau, mais être interrompu par du langage HTML, ce qui limite la connaissance nécessaire de ce langage. Le résultat de ce script sera :



La récupération se fera plus simplement dans la mesure où la variable aura toujours une valeur, du moins si un défaut a été fixé :

```
<?php
$choix=$_POST["choix"];
?>
```

Les cases à cocher

Les cases à cocher, correspondant à la balise "input" et au type "checkbox", permettent des choix multiples. Ici contrairement aux cases d'options, il faudra donner un nom différent à chaque case. Seules les cases cochées seront envoyées au serveur. Si l'on a un nombre suffisant de choix, il est possible de stocker les valeurs dans un tableau PHP, dans ce cas le nom des cases à cocher sera un nom de tableau c'est à dire suivi de [], le même pour l'ensemble des cases. L'exemple précédent avec cette fois ci des cases à cocher.

```
Que préférez vous?</p>
<?php
// tableau contenant les intitulés mis en valeurs
$lieux=array("Ville","Montagne","Mer","Campagne","Etranger");
for($i=0;$i<5;$i++)
{
?>
<input type="checkbox" value=
```




```

<?php
echo $lieux[$i];
?>
name="choix[]">
<?php
//écriture de l'intitulé sur la page HTML
echo $lieux[$i];
}
//fermeture de la boucle for
?>
</p>

```

Le résultat de ce script sera :



Que préférez vous?

☐ Ville ☐ Montagne ☐ Mer ☐ Campagne ☐ Etranger

Si les noms sont associés à un tableau, on pourra pour le traitement des résultats utiliser une boucle foreach :

```

<?php
if(isset($_POST["choix"]))
{
    $choix=$_POST["choix"]
    foreach($choix as $elt)
        traiter elt
    }
?>

```

Les zones de liste déroulante

Ces zones, annoncées par la balise "select" permettent à l'utilisateur de choisir parmi une liste de choix prédéfini, une valeur peut être présélectionnée. Leur création peut se faire soit directement avec un outil de création de page Web, soit par programme PHP sur un schéma ressemblant à celui vu précédemment, mais dans ce cas, après la balise select de définition de la liste, chacune des options possibles est déclarée dans une balise option sans propriété de nom puisque la valeur sera donnée à la balise "select". Ce qui conduit au programme suivant :

```

<form method=POST action=liste.php>
Que préférez vous?<select name="maliste">
<?php
// tableau contenant les intitulés mis en valeurs
$lieux=array("Ville","Montagne","Mer","Campagne","Etranger");
for($i=0;$i<5;$i++)
{
    ?>
    <option value=
    <?php
    echo $lieux[$i]. " ";
    // par défaut le premier choix
    if($i==1){
    ?>
        selected
    <?php
    //fin du if
    ;}
    ?>
    > Remarque fermeture de la balise ouvrante option
    <?
    //affichage de la valeur

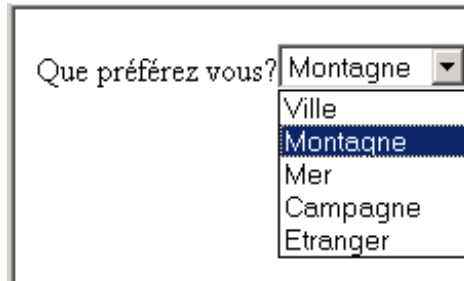
```

```

echo $lieux[$i];
?>
</option>
<?php
} //fermeture de la boucle for
?>

```

Ce qui donnera l'affichage suivant :



Pour récupérer, sur le serveur, la valeur choisie par l'utilisateur on écrira :

```

<?php
if(isset($_POST["maliste"]))
$monchoix=$_POST["maliste"];
?>

```

La zone cachée

Les informations entrées par le client ne peuvent à priori être utilisées que par le fichier correspondant à la propriété **action** du formulaire. Si on doit utiliser ces informations dans un autre fichier, en complément de nouvelles informations, il ne faut évidemment pas les redemander au client, une méthode consiste à les mettre dans une zone cachée du nouveau formulaire soumis au client. Cette zone est annoncée par une balise "input" de type "hidden", sa valeur sera pratiquement toujours donnée par programme :

```

<input type="hidden" name="mavar" value=
<?php echo $mavar;
?>
>

```

remarque : fermeture de la balise input

La récupération dans le fichier action se fait comme pour les zones de texte.

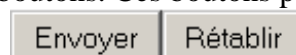
La saisie d'un mot de passe

Pour saisir un mot de passe, plutôt que d'utiliser une zone de texte, on utilisera une zone qui ne fait pas apparaître en clair le texte tapé par l'utilisateur. Ceci correspond à une balise input de type "password". A l'écran n'apparaissent que des astérisques. Pour cela il suffit de créer une zone de texte et de changer le type dans la balise. On récupère le mot de passe comme pour une zone de texte.

Attention, le mot de passe n'est pas codé mais est transmis en clair

Les boutons

Pour déclencher l'envoi au serveur de la requête de l'URL réponse il est nécessaire de mettre dans le formulaire un ou plusieurs boutons. Ces boutons peuvent être des boutons classiques :



ou des images :



Pour les créer, on utilisera un outil de création de page Web en insérant le contrôle correspondant à la balise "input" de type "submit" pour les boutons standards ou "image" pour

les images. Ne pas oublier de donner un nom à ces contrôles. Pour les boutons illustrés, on aurait les balises suivantes :

```
<input type="submit" value="Envoyer" name="Envoi">
<input type="submit" value="Rétablir" name="Retab">
<input type="image" src="poubelle.gif" name="poub" width="23" height="30">
```

Dans la page de réponse on utilisera le nom de ces contrôles pour savoir lequel a été cliqué.

Attention pour un contrôle image, dans la variable `$_POST`, la clé correspondante est le nom du contrôle suivi de `"_x"`.

Pour traiter dans le fichier de réponse les différents cas, il faudra avoir un code tel que :

```
<?php
// bouton envoyer
if(isset($_POST["Envoi"]))
{ traitement de l'envoi }
// bouton rétablir
elseif (isset($_POST["Retab"]))
{traitement de rétablir}
//image poubelle
elseif (isset($_POST["poub_x"])) // attention au _x
{traitement poubelle}

?>
```

La gestion des cookies

Les cookies servent à enregistrer des données sur l'ordinateur du visiteur. Celles-ci peuvent ensuite être récupérées dans l'ensemble des pages d'un site. Un exemple simple d'utilisation de cookie : la date de la visite de la page. Il n'est pas recommandé de stocker des informations importantes dans un cookie, dans la mesure où le navigateur peut être configuré pour refuser les cookies et dans la mesure où l'utilisateur peut supprimer les cookies de son poste client soit directement soit en installant un nouveau navigateur.

La portée d'un cookie est déclarée dans la réponse HTTP envoyée par le serveur web. Celle-ci renferme des informations : information de chemin d'accès (ex : /chemin), information d'expiration (ex : 01/12/2000 12:00:00), information de domaine (ex : www.monsite.com), un paramètre de sécurité.

Les informations sur le domaine permettent de restreindre l'accès au cookie. Pour définir le cookie sur l'ensemble des hôtes du domaine, il suffit d'ajouter un point devant le nom du domaine : ".www.monsite.com". Les informations sur l'expiration, cela permet de restreindre la durée de vie du cookie, il est donc possible de déclarer un cookie qui ne restera sur l'ordinateur du visiteur qu'une certaine durée.

Le paramètre de sécurité permet de restreindre l'envoi du cookie, ce qui permet de ne faire envoyer le cookie qu'aux serveurs sécurisés (HTTPS). Si le paramètre est désactivé, le cookie sera envoyé dans tous les cas.

Cookie à valeur unique

Pour envoyer un cookie, il faut utiliser la fonction `setcookie()` de PHP :

```
setcookie(nom,valeur,expiration,chemin,domaine,sécurisé);
```

Si le paramètre expiration est omis, le cookie disparaîtra à la fin de la session, soit en général quand le navigateur est fermé..

Si le paramètre chemin est omis, c'est le chemin du fichier envoyant le cookie qui est utilisé.

Si le paramètre domaine est omis c'est le domaine envoyant le cookie qui est utilisé.

Les arguments chemin et domaine sont utilisés pour déterminer s'il y a lieu de fournir le cookie à un script : seul les scripts du chemin du domaine seront autorisés à recevoir le cookie. C'est

pour cela qu'il est spécifié que domaine doit contenir au moins 2 points, pour éviter l'envoi à des génériques tels que ".com".

Le paramètre sécurisé vaut 1 pour l'envoi du cookie à travers un protocole sécurisé, 0 sinon et par défaut.

La fonction setcookie ne renvoie pas d'indication sur l'acceptation du cookie par le client, il vous faut le vérifier à la requête suivante.

Exemple : envoi d'un cookie contenant la date du jour et expirant dans 10 jours :

```
< ?php
$visite=date("d/m/y");
setcookie("visite", $visite, time()+60*60*24*10);
?>
```

Cookie à valeurs multiples

Il est aussi possible d'envoyer un cookie a valeurs multiples, pour cela, on traite le nom du cookie comme un tableau.

Envoi d'un cookie contenant la date de la visite et la page d'où le visiteur venait

```
< ?php
$visite=date("d/m/y");
setcookie("autre[0]", $visite, time()+60*60*24*10);
$origine=$_SERVER["HTTP_REFERER"];
setcookie("autre[1]", $origine, time()+60*60*24*10);
?>
```

Définir la date d'expiration

Lors de la création d'un cookie, on définit la durée de vie de celui-ci, cela permet de garder par exemple pour un temps donné certaines information. On peut soit définir cette date d'expiration par : une durée fixe depuis la création, en utilisant la fonction time() :

```
$nbjours=10;
```

```
$expiration=time()+24*60*60*$nbjours;
```

soit une date et une heure précise à laquelle le cookie expire, on utilisera alors la fonction mktime qui retourne le nombre de secondes écoulées depuis le 1 janvier 1970.

```
mktime=(heure,minute,seconde,jour,année,heure d'hiver)
```

tous les paramètres sont des entiers, le dernier paramètre vaut 1 s'il l'heure d'hiver est utilisée, 0 sinon, -1 si on laisse PHP le déterminer.

Exemple : envoi d'un cookie expirant le 1 janvier 2007 à 9heures :

```
<?php
$expiration = mktime(9,0,0,1,1,2007,1);
setcookie("cookie", "essai", $expiration, "/");
?>
```

Récupération d'un cookie

Pour récupérer un cookie, il suffit d'utiliser la variable superglobale \$_COOKIE qui est prédéfinie. Dans notre exemple précédent, pour récupérer la date de la dernière visite nous utiliserons le petit script suivant :

```
<?php
if ($_COOKIE["visite"]!="")
{
    $t= $_COOKIE["toto"];
    echo $t; }
else{echo "ce site demande d'accepter les cookies";}
?>
```

Si le cookie est à valeurs multiples, la variable `$_COOKIE[]` associée est alors un tableau.

Suppression d'un cookie

Pour supprimer un cookie, il vous suffit de déclarer le cookie sans mettre de valeur :

```
<?php
setcookie("visite");
?>
```

Erreur à éviter

Les cookies sont envoyés au navigateur dans les entêtes HTTP, ce qui oblige à les envoyer avant toute autre donnée.

Le script suivant :

```
<?php
echo "Bonjour";
setcookie("essai", "essai_valeur");
?>
```

va renvoyer :

Bonjour *Warning: SetCookie called after header has been sent in planco.php on line 3*

Il faut donc que l'envoi de cookie se fasse sur les premières lignes. Le script suivant provoquera la même erreur (les numéros de ligne sont entre parenthèses :

```
(1) :
(2) : < ?php
(3) : setcookie("essai", "essai_valeur");
(4) : ?>
```

Explication : ci-dessus, à la ligne 1, nous avons laissé un espace, donc cet espace étant avant la balise `< ?`, il sera renvoyé tel quel au navigateur. Cet espace empêche la création du cookie.

Tandis que le code suivant est correct :

```
(1) < ?php
(2) :
(3) : setcookie("essai", "essai_valeur");
(4) : ?>
```

Les variables session

Une autre façon de conserver des informations de page et d'utiliser la session, et la variable surperglobale `$_SESSION` qui y est attachée. On peut considérer que la session correspond à un cookie stockée sur le serveur, c'est donc un fichier dans lequel on peut stocker des informations avec plus de sécurité. Quand un visiteur se connecte à votre site, un identifiant lui est assigné, et vous permet donc de reconnaître d'où viennent les requêtes. La durée de vie d'une session est limitée : après un certain temps d'inactivité fixé par le serveur (généralement 30 minutes) et/ou par la fermeture du navigateur sur le client.

L'identifiant de session (SID) est envoyé au navigateur sous forme de cookie, donc dans l'entête HTTP. Comme le client peut refuser les cookies, PHP crée une constante appelée `SID` qui permet d'envoyer l'identifiant de session dans une adresse URL.

On utilisera ensuite le tableau `$_SESSION` pour stocker les informations voulues.

La fonction `session_start()`

Cette fonction doit être appelée sur toutes les pages du site, elle permet de démarrer une session ou d'appeler la session existante (récupérer son identifiant). Il est nécessaire que cette fonction soit appelée avant tout envoi au navigateur : pas de HTML, pas de fonction `echo`.

Une bonne pratique est de la mettre en première ligne de chaque page.

Utilisation du tableau \$_SESSION

Une fois la session démarrée, le tableau \$_SESSION peut être utilisé pour stocker des informations utilisables sur toutes les pages, sans les passer par un champ caché de formulaire.

Premier exemple, après avoir rempli un formulaire demandant un pseudo, on stocke ce pseudo dans une variable globale, ainsi que son heure de connexion. L'ensemble du script de la page est donné :

```
<?php
    session_start();
?>
</html>

<head>
</head>
<body bgcolor="#FFFFFF">
<?php
if(isset($_POST["pseudo"]) )
{
//le formulaire a été rempli
$_SESSION["pseudo"]=$_POST["pseudo"];
$_SESSION["arrivée"]=time() ;
?>
<p><a href="verif.php">Vers vérification</a></p>
<?php
}
else
{
//le formulaire est affiché
?>
<form method="POST" action="crevar.php">
    <p>pseudo<input type="text" name="pseudo" size="20">
    <input type="submit" value="Envoyer" name="B1"></p>
</form>
<?php
} //fin du else
?>
</body>

</html>
```

Tout ce qui est en italique concerne le code HTML. Ce fichier nommé "crevar.php", gère à la fois le formulaire et le stockage des variables session.

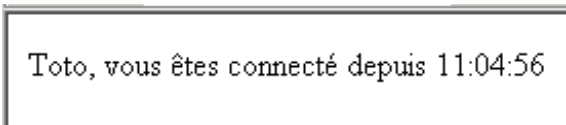
Après avoir démarré ou appelé la session, on vérifie que le pseudo a été rentré, si oui on effectue le stockage du pseudo et de l'heure de connexion puis on crée un lien (en HTML) vers un fichier de vérification (verif.php), sinon on affiche le formulaire. Voici les deux écrans correspondant

| Formulaire | Appel de la page de verif |
|---|--|
| <p>pseudo <input type="text"/> <input type="submit" value="Envoyer"/></p> | <p>Vers vérification</p> |

Voici maintenant le script de vérification :

```
<?php
session_start();
?>
<html>
<head>
</head>
<body bgcolor="#FFFFFF">
<?
    // renvoie le pseudo et l'heure d'arrivée
    echo $_SESSION["pseudo"].", vous êtes connecté depuis ".
    date("h:m:s",$_SESSION["arrivée"]);
?>
</body>
</html>
```

et l'écran correspondant :



Toto, vous êtes connecté depuis 11:04:56

Pour supprimer une variable session, on utilisera la fonction **unset()**, exemple :

```
unset($_SESSION["pseudo"]);
```

Il est bien sûr, possible de créer des variables session qui sont des tableaux, par exemple pour un panier on pourrait, par exemple, utiliser un tableau à trois dimensions :

```
$_SESSION["panier"][$code_article]["quantité"]=$quant;
$_SESSION["panier"][$code_article]["prix"]=$prix;
....
```

Pour enlever du panier l'article de code stocké dans la variable \$code_article, il suffira alors de l'instruction :

```
unset($_SESSION["panier"][$code_article]);
```

Destruction d'une session

Il peut être utile de permettre à l'utilisateur de se délogger, dans ce cas il faut détruire les variables session avant de mettre fin à la session. Pour mettre fin à la session, on utilise la fonction `session_destroy()`. Voici le script correspondant :

```
<?php
session_start();
?>
<?php
    $_SESSION=array();
    session_destroy();
?>
```

Remarquez que l'on ne peut utiliser **unset** sur la variable `$_SESSION`, puisque c'est une variable superglobale, on détruit uniquement son contenu avec un tableau vide.

Utilisation d'une base de donnée

Nous n'entrerons pas ici dans le détail de la syntaxe SQL, mais simplement nous indiquerons les principales fonction PHP utile pour la gestion d'une base de donnée. Nous nous limiterons de plus à la base de données relationnelle Mysql. Toutes les fonctions utilisées commencent par mysql_.

Connexion au serveur Mysql

Avant toute opération sur une base de données, il est nécessaire de se connecter à cette base. Dans un premier temps, on se connecte sur le serveur de base de données, puis ensuite on sélectionne la base de données sur la quelle on veut travailler.

La connexion au serveur de base de données se fait par la fonction mysql_connect, dont la syntaxe simplifiée est la suivante :

mysql_connect(hôte,utilisateur,mot de passe);

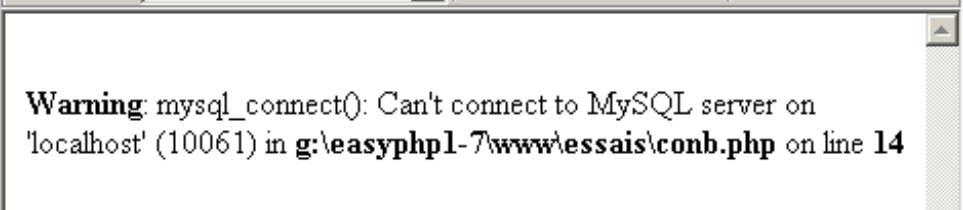
les trois arguments sont de type chaîne de caractères.

L'hôte est l'adresse d'hébergement de la base de données, par défaut c'est "localhost".

Utilisateur est le nom de l'utilisateur, par défaut le nom utiliser "root":

```
$icon=mysql_connect("localhost","root","");
```

La fonction **mysql_connect()** retourne un entier qui sera utilisé dans d'autres fonctions, si PHP ne réussit pas à se connecter la fonction retourne FALSE. Dans le cas de non-connexion on obtient un message du type :

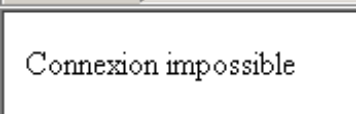


Warning: mysql_connect(): Can't connect to MySQL server on 'localhost' (10061) in g:\easyphp1-7\www\essais\conb.php on line 14

Il existe deux moyens pour éviter ce message, le premier consiste à utiliser la fonction die(message) qui arrête le script en envoyant un message indépendant de l'erreur de connexion et à faire précéder l'usage de la fonction par un "@" qui bloque les messages d'erreur.

```
@$icon=mysql_connect("localhost","root","") or die("Connexion impossible");
```

En cas d'échec à la connexion on aura le résultat suivant :




Connexion impossible

Si la connexion se passe correctement le script exécutera les instructions suivantes. Toutefois dans ce cas on ne sait pas ce qui provoque l'échec de la connexion. Pour être plus précis, il est possible d'utiliser, après un test, la fonction mysql_error() qui retourne le message précis de l'erreur puis la fonction exit qui arrête l'exécution :

```
<?php
if(! @ $icon=mysql_connect("localhost","root","jenesaispas"))
{echo mysql_error();
exit();}
instructions en cas de connexion;
?>
```

conduira au résultat suivant :



Accès refusé pour l'utilisateur: 'root@localhost' (mot de passe: OUI)

La connexion prend fin à l'exécution du script, il est donc nécessaire de se reconnecter dans chaque page.

Choix de la base de données

Après la connexion au serveur, il faut sélectionner la base de données sur la quelle on veut travailler. Ceci se fait avec la fonction :

mysql_select_db(base,identifiant de connexion);

Base est le nom de la base de données, connexion est l'identifiant retourné par la fonction mysql_connect. Le résultat est un booléen TRUE si la base est trouvée, FALSE sinon. Le script complet de connexion à la base pourra être le suivant :

```
<?php
if(! @ $icon=mysql_connect("localhost","root",""))
{echo mysql_error();
exit();}
mysql_select_db("CDtheque",$icon) or die("Base inconnue");
?>
```

Exécution d'une requête

Pour exécuter une requête, on utilise sous sa forme simplifiée la fonction :

mysql_query(requête);

Dans le cas d'une requête SELECT, le retour de la fonction est un identifiant du résultat, dans les cas des requêtes INSERT, DELETE, UPDATE le retour de la fonction est un booléen indiquant les succès de l'opération.

Exemples :

1)Requête de mise à jour : on change le prix du produit de code donné \$icode :

```
<?php
$code=5;
$nprix=25;
$myquer='UPDATE `cd` SET `prix` = '.$nprix.' WHERE `num_cd` = '.$code.';';
if(!mysql_query($myquer))echo "Mise à jour impossible";
?>
```

2)Requête d'ajout d'un compositeur et de sa nationalité, le champ commentaire n'est pas rempli. Attention dans ce cas les valeurs, qui sont des chaînes de caractères doivent être guillemets :

```
<?php
$nom="Puccini";
$nat="Italien";
$myquer='INSERT INTO `compositeurs` ( `Nom_C` , `Nationalité` ) VALUES
("'.$nom.'","'.$nat.'")'; //noter les " pour les chaînes!
if(!mysql_query($myquer))echo "Ajout impossible";
?>
```

3)Sélection de tous les disques d'un éditeur donné :

```
<?php
$editeur="Deutsche Grammophon";
$myquer='SELECT * FROM `cd` WHERE `n_editeur` LIKE "'.$editeur.'"'; //attention aux "
$result=mysql_query($myquer);
?>
```

Il nous faut maintenant analyser le résultat de la requête.

Analyse du résultat d'une requête de sélection

On peut considérer que le résultat de la requête SELECT pointe sur une table. Pour analyser ce résultat, la démarche consiste à regarder chaque ligne de la table à l'aide de fonctions PHP. Pour ce faire, on utilisera une boucle sur l'une des deux fonctions

mysql_fetch_row(résultat de la requête);

qui, à chaque appel, donne un tableau, correspondant à une nouvelle ligne de la table, indicé à partir de 0 pour chacun des éléments de la ligne, ou la fonction :

```
mysql_fetch_array(résultat de la requête);
```

qui, elle, donne un tableau associatif de la ligne où la clé est soit le nom du champ, soit comme pour la fonction précédente un indice commençant à 0. Cette dernière fonction rend les scripts plus lisibles si on utilise la clé.

Quand on arrive à la fin de la table, la fonction retourne un tableau vide qui peut être converti en booléen FALSE, d'où la structure d'analyse du résultat d'une requête :

```
While(enr=mysql_fetch_array($result)){traitement}
```

Illustrons cette analyse en affichant le titre du CD, sa référence, son compositeur dans le cas du paragraphe précédent.

Avec la fonction `mysql_fetch_row` :

```
<?php
    $editeur="Deutsche Grammophon";
    $myquer='SELECT * FROM `cd` WHERE `n_editeur` LIKE "'.$editeur.'";
    $result=mysql_query($myquer);
    while($enr= mysql_fetch_row($result))
        echo "titre:$enr[1] Référence:$enr[5] Compositeur:$enr[3]<br>";
?>
```

Avec la fonction `mysql_fetch_array`, on pourrait utiliser le même script, mais en utilisant les clés, il faut alors utiliser l'opérateur de concaténation puisque ces clés sont des chaînes de caractères :

```
<?php
    $editeur="Deutsche Grammophon";
    $myquer='SELECT * FROM `cd` WHERE `n_editeur` LIKE "'.$editeur.'";
    $result=mysql_query($myquer);
    while($enr= mysql_fetch_array($result))
        echo "titre:".$enr["titre_cd"]." Référence:".$enr["Reference"]."
Compositeur:".$enr["n_compo"]."<br>";
?>
```

Dans les deux cas le résultat sera le suivant :

```
titre:Chopin, Concerto pour piano numéro 2 Référence:410507-2 Compositeur:Chopin
titre:Verdi, il trovatore Référence:415285-2 Compositeur:Verdi
titre:Johannes Brahms, Hungarian Dances Référence:410615-2 Compositeur:Brahms
titre:Carmen, Georges Bizet Référence:419636-2 Compositeur:Bizet
```

Remarque : les noms des champs, ou l'ordre des champs, ne correspondent pas nécessairement à la base qui vous a été donnée.

Il existe aussi une fonction donnant le nombre de lignes dans la table résultat de la requête, ce qui est utile pour vérifier qu'il existe au moins un enregistrement correspondant à celle ci :

```
$nblignes=mysql_num_rows($result);
```

Gestion des fichiers en PHP

Puisque l'interpréteur est sur le serveur, la lecture ou l'écriture d'un fichier ne peut se faire que sur les répertoires du ou des serveurs WEB où vous disposez des droits suffisants. Dans une première partie, nous verrons comment accéder à un fichier sur le serveur, dans la seconde partie nous verrons comment télécharger un fichier à partir du poste client.

Ouverture d'un fichier

La première opération à faire est d'ouvrir le fichier sur le serveur, c'est à dire lui associer une variable ressource (un handle), qui gérera les flux physiques, cette variable sera utilisée dans les autres fonctions d'opérations sur ce fichier. La fonction associée à l'ouverture d'un fichier est la fonction ***fopen***(*nom du fichier, mode d'ouverture*) qui retourne faux en cas d'erreur, le handle sinon.

Le mode d'ouverture est une chaîne de caractère indiquant le traitement que l'on envisage sur le fichier :

- **r** : ouverture en lecture seulement
- **w** : ouverture en écriture seulement (la fonction crée le fichier s'il n'existe pas)
- **a** : ouverture en écriture seulement avec ajout du contenu à la fin du fichier (la fonction crée le fichier s'il n'existe pas)
- **r+** : ouverture en lecture et écriture
- **w+** : ouverture en lecture et écriture (la fonction crée le fichier s'il n'existe pas)
- **a+** : ouverture en lecture et écriture avec ajout du contenu à la fin du fichier (la fonction crée le fichier s'il n'existe pas)

Remarque si l'on veut éviter d'écraser des informations contenues dans un fichier, où l'on veut écrire, il est recommandé d'utiliser les modes **a** ou **a+**.

On pourra donc écrire par exemple :

```
<?php
$fichier = nom de l'URL;
$type = type d'ouverture;
@$fh = fopen($fichier,"r") or die( "Impossible d'ouvrir le fichier");
```

Traitement

```
fclose($fh);
?>
```

Fermeture d'un fichier

Un fichier ouvert avec ***fopen*** doit être fermé avec l'a fonction ***fclose***(*handle*) (cf ci-dessus)

Lecture d'un fichier

PHP donne plusieurs possibilités de lecture d'un fichier, la lecture du fichier en une seule fois, la lecture ligne par ligne d'un fichier texte ou la lecture octet par octet. Il existe aussi des fonctions associées à des types spécifiques de fichiers texte : les fichiers csv à séparateurs et les fichiers HTML.

Lecture complète d'un fichier

La fonction ***fread***(*handle, nombre maximum de caractères*) retourne le contenu total d'un fichier dans une chaîne de caractères. Il est possible, dans le cas d'un fichier se trouvant dans le même domaine, d'utiliser la fonction ***filesize***(*nom de fichier*) pour déterminer le nombre maximum de caractères à lire.

Exemple, lecture d'un fichier se trouvant dans le répertoire "Monautresite" du même domaine :

```
<?php
$fichier="../Monsite_1/Berlioz.jpg";
@$fh=fopen($fichier,"r") or die( "Impossible d'ouvrir le fichier");

$pf=fread($fh,filesize($fichier));
Traitement

fclose($fh);
?>
```

Remarques :

1. Il est possible de lire en entier un fichier sans utiliser de handle avec la fonction `file(nom du fichier)` qui retourne les octets dans un tableau. Chaque élément du tableau correspond à une ligne du fichier. Il faut dans ce cas s'assurer avant d'utiliser la fonction ***file()*** que le fichier existe bien par la fonction ***file_exists(nom du fichier)***. Exemple :
2. Si l'on veut lire un fichier et uniquement l'afficher (par exemple une image), on peut utiliser la fonction `fpasssthru(handle)`. En fait cette fonction permet d'afficher le reste du fichier à partir du point où en est la lecture. Cette fonction retourne le nombre d'octets lus.

Exemple :

```
<?php
$fichier="../Monsite_1/Berlioz.jpg";
@$fh=fopen($fichier,"r") or die( "Impossible d'ouvrir le fichier");

fpasssthru($fh);

fclose($fh);
?>
```

Lecture ligne par ligne d'un fichier texte

La lecture ligne par ligne se fait à l'aide de la fonction ***fgets(handle,nb max de caractères)***. Pour lire plusieurs ligne d'un fichier, on utilisera une structure répétitive. Attention cependant à ne pas dépasser la fin de fichier, pour cela il existe une fonction booléenne ***feof(handle)*** retournant vrai en cas de fin de fichier rencontrée. On privilégiera donc la structure `while`, par exemple pour lire 10 lignes au maximum dans un fichier on utilisera un script tel que :

```
<?php
$fichier="montest.txt";
@$fh=fopen($fichier,"r") or die( "Impossible d'ouvrir le fichier");
$i=0;
while($i<10 && !feof($fh))
{
    $ligne=fgets($fh,1000);
    Traitement
    $i++;
}
echo $i." lignes ont été lues";
fclose($fh);
?>
```

Lecture d'un fichier csv

Un fichier csv est un fichier texte dont les données sont séparées par un séparateur. Par défaut ce séparateur en PHP est supposé être la virgule. Par exemple un fichier pour un publipostage, contenant le nom, la ville et le mail des individus :

Dupond, Amiens, jdupond@free.fr

Martin, Paris 15, almart@wanadoo.fr

.....

Chaque ligne est lue dans un tableau dont les éléments sont les données individuelles.

La fonction utilisée est la fonction *fgetcsv(handle, nbmax, séparateur)*, nbmax représente le nombre maximum de caractères lus sur une ligne, séparateur indique le caractère séparateur de données, s'il est omis PHP prendra la virgule comme caractère séparateur. Prenons comme exemple le fichier précédent, nous allons introduire dans le programme deux fonctions de chaînes de caractères qui peuvent être utiles:

La fonction *trim(chaîne)* qui élimine les espaces en début et fin de chaîne

La fonction *strpos(chaîne, sous-chaîne, début)* qui retourne la position d'une sous chaîne dans une chaîne à partir du caractère en position début. Cette fonction retourne FALSE si la sous-chaîne n'est pas trouvée, sinon la position relative par rapport à début en commençant par 0.

Ici cela nous servira à vérifier que l'adresse mail est acceptable.

Voici le script :

```
<?php
$fichier="montest.txt";
@$fh=fopen($fichier,"r") or die( "Impossible d'ouvrir le fichier");
$i=0; // pour repérer la ligne
//lecture ligne par ligne de tout le fichier
while( !feof($fh))
{
    $i++;
    $ligne=fgetcsv($fh,1000);
    $nom=trim($ligne[0]);
    $ville=trim($ligne[1]);
    $mail=trim($ligne[2]);
    // il faut un @
    $premat=strpos($mail,"@");
    //mais pas deux
    $deuxat=strpos($mail,"@",$premat+1);
    if ($premat && !$deuxat)
    {
        Traitement
    }
    else
    {
        echo "adresse mail invalide pour la ligne $i";Traitement
    }
}
fclose($fh);
?>
```

Remarques sur le script :

- \$premat donne la position du premier @
- \$deuxat cherche un deuxième @, donc la position de début est \$premat+1
- Pour que l'adresse soit acceptable, il faut que la première recherche soit fructueuse (donc un résultat transformé en booléen TRUE) et que la deuxième retourne la valeur FALSE.

Ecriture dans un fichier

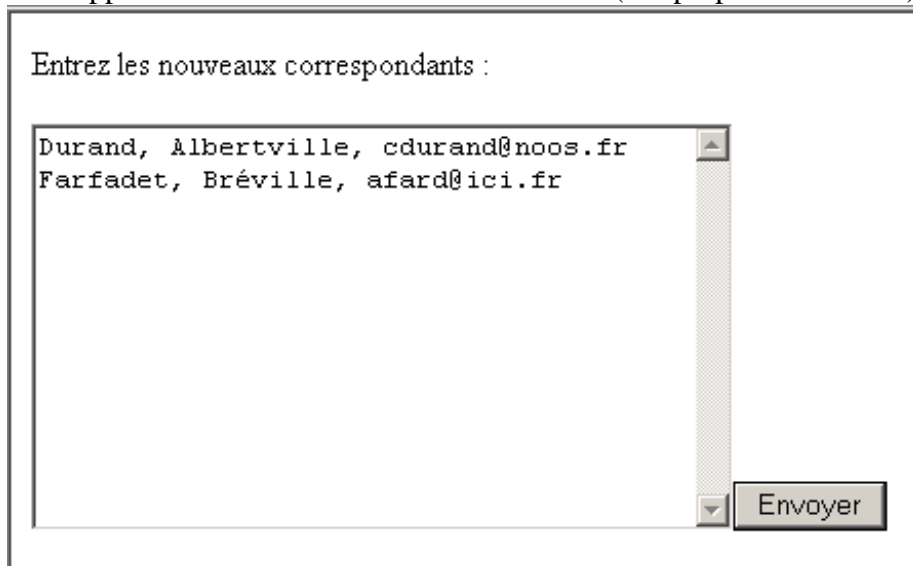
L'écriture d'une chaîne de caractères dans un fichier se fait avec la fonction

fputs(handle, chaîne). Pour le passage à la ligne on envoie la chaîne "\n".

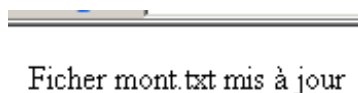
Il est possible de faire saisir le texte dans une zone "espace de texte" (textarea) qui permet de saisir plusieurs lignes, et dans le fichier correspondant à l'action d'écrire ces lignes dans un fichier. Voici un script correspondant dans un fichier nommé "fecri.php":

```
<?php
if (isset($_POST["letexte"]))
{
    $fichier="mont.txt";
    @$fh=fopen($fichier,"w") or die( "Impossible d'ouvrir le fichier");
    fputs($fh,$_POST["letexte"]);
    echo "Fichier $fichier mis à jour";
}
else
{
    // le formulaire en HTML
    ?>
        Entrez les nouveaux correspondants :
        <br>
        <form method="POST" action="fecri.php">
            <p><textarea rows="12" name="letexte" cols="40">
            </textarea><input type="submit" value="Envoyer" name="B1"></p>
        </form>
    <?php
    //fin du else
    }
    ?>
```

Lors du premier appel du fichier on obtient l'écran suivant (rempli par l'utilisateur):



Puis après envoi :



Fichier mont.txt mis à jour

Autres fonctions utiles pour les fichiers ou répertoires du serveur

PHP fournit de nombreuses fonctions permettant de faire des tests ou d'obtenir des informations sur les fichiers. En voici quelques unes:

- ***is_dir()*** permet de savoir si le fichier dont le nom est passé en paramètre correspond à un répertoire

is_dir(Nom du fichier);

La fonction *is_dir()* renvoie TRUE s'il s'agit d'un répertoire, FALSE dans le cas contraire

- ***is_executable()*** permet de savoir si le fichier dont le nom est passé en paramètre est exécutable

is_executable(Nom du fichier);

La fonction *is_executable()* renvoie TRUE si le fichier est exécutable, FALSE dans le cas contraire

- ***is_file()*** permet de savoir si le fichier dont le nom est passé en paramètre ne correspond pas à un répertoire.

is_file(Nom du fichier);

La fonction *is_file()* renvoie TRUE s'il s'agit d'un fichier, FALSE dans le cas contraire

- ***filetime()*** permet de connaître la date et l'heure du dernier accès au fichier.

filetime(Nom du fichier)

La fonction *filetime()* retourne la date et l'heure du dernier en nombre de secondes depuis le 1 janvier 1970, il vous faut donc utiliser la fonction *date()* pour visualiser facilement ce résultat.

Exemple :

```
<?php
$fichier="montest.txt";
$date=filetime($fichier);
$date=date("l d/m/y H:m:s",$date);
echo "le dernier accès a eu lieu le : $date";
?>
```

- ***filemtime()*** permet de connaître la date et l'heure de la dernière modification du fichier.

filemtime(Nom du fichier)

La fonction *filemtime()* retourne la date et l'heure du dernier en nombre de secondes depuis le 1 janvier 1970, il vous faut donc utiliser la fonction *date()* pour visualiser facilement ce résultat.

- ***unlink()*** permet de supprimer un fichier.

unlink(Nom du fichier)

Chargement d'un fichier à partir du client

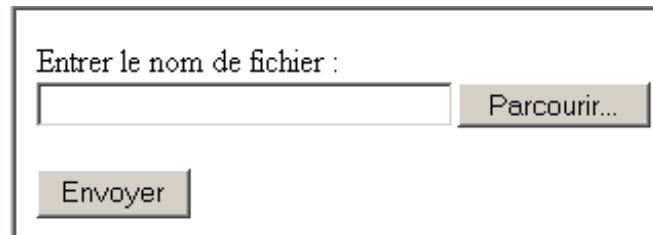
Le chargement d'un fichier à partir du client se fait en deux temps : tout d'abord on crée un formulaire pour demander le nom complet du fichier, lors de l'exécution du formulaire le fichier est stocké dans un répertoire temporaire du serveur, le script de réponse traite alors le fichier temporaire ou le stocke dans un répertoire du domaine. Attention à la fin de l'exécution du script de réponse, le fichier est effacé du répertoire temporaire.

Le formulaire se saisie du nom du fichier client

Ce formulaire contient un objet input de type files qui permet de parcourir les disques accessibles du serveur pour localiser le fichier. Le script HTML est le suivant :

```
<form method="POST" enctype="multipart/form-data" action="traitfic.php">
  <p>Entrer le nom de fichier :
  <input type="file" name="fichier" size="30"></p>
  <input type="submit" value="Envoyer" name="B1">
</form>
```

et l'apparence :



Entrer le nom de fichier :

Parcourir...

Envoyer

Remarque : vous pouvez limiter la taille maximum du fichier téléchargé en ajoutant au formulaire de saisie un champ caché fixant la constante `MAX_FILE_SIZE`. Ici par exemple pour limiter à 100Koctets :

```
<input type="hidden" name="MAX_FILE_SIZE" value="100000">
```

Le script de réponse

Le script de réponse reçoit dans la variable superglobale `$_FILES` un tableau contenant les informations suivantes :

- `$_FILES["fichier"]["name"]` : fournit le nom d'origine du fichier.
- `$_FILES["fichier"]["type"]` : fournit le type MIME du fichier.
- `$_FILES["fichier"]["size"]` : fournit la taille en octets du fichier.
- `$_FILES["fichier"]["tmp_name"]` : fournit le nom temporaire du fichier.
- `$_FILES["fichier"]["error"]` : fournit le code d'erreur associé au téléchargement.

La taille maximum d'un fichier téléchargeable est fixée par la directive `upload_max_filesize` du fichier **php.ini** et est par défaut de 2Mo.

Les codes erreurs sont des constantes définies par PHP, toutefois il est plus simple d'utiliser la fonction `is_uploaded_file()` qui retourne `TRUE` si le téléchargement s'est bien passé, `FALSE` sinon :

```
is_uploaded_file($_FILES["fichier"]["tmp_name"])
```

C'est le nom du fichier temporaire qui est utilisé comme paramètre dans cette fonction booléenne.

Si le chargement s'est passé correctement, pour sauvegarder le fichier sur le serveur on utilisera la fonction `move_uploaded_file()` qui a deux paramètres le nom du fichier temporaire, le nom du fichier destination :

```
move_uploaded_file(fichier temporaire, fichier destination)
```

qui retourne `TRUE` si le déplacement se passe bien, `FALSE` sinon. Cette fonction ne peut-être utilisée qu'avec la méthode POST.

Voici le script de réponse :

```
<?php
if (isset($_FILES["fichier"]))
{
    // le téléchargement est-il OK?
    if(is_uploaded_file($_FILES["fichier"]["tmp_name"]))
    {
        // déplacement du fichier dans le répertoire images avec le nom d'origine
        $destination="images/".$_FILES["fichier"]["name"];
        //on garde éventuellement la taille
        $taille=$_FILES["fichier"]["size"];
        move_uploaded_file($_FILES["fichier"]["tmp_name"], $destination);
        traitement éventuel du fichier destination
    }
    else
    {echo "Le téléchargement a échoué";}
}
?>
```


Téléchargement de plusieurs fichiers

Comme pour les autres contrôles input, on peut utiliser un tableau pour sélectionner plusieurs fichier à télécharger. Les éléments de la variable \$_FILES["fichier"] sont alors des tableaux indicés à partir de 0, \$_FILES["fichier"]["name"][i], \$_FILES["fichier"]["tmp_name"][i] etc. Voici un exemple permettant de télécharger 5 fichiers au maximum, de déplacer ces fichiers dans un répertoire et de conserver dans un tableau leurs caractéristiques :

```
<body>
  <?php
    //définir le nombre max de fichiers
    DEFINE("maxfic",5);
    // réponse au bouton d'envoi
    if(isset($_FILES["fichier"]))
    {
        $fichiers=$_FILES["fichier"];
        $i=0; //compteurs de fichiers réellement chargés
        for($cle=0;$cle<maxfic;$cle++)
        {
            if(is_uploaded_file($fichiers["tmp_name"][$cle]))
            {
                $nom[$i]=$fichiers["name"][$cle];
                $destination="images/".$nom[$i];
                $taille[$i]=$fichiers["size"][$cle];
                $type[$i]=$fichiers["type"][$cle];
                move_uploaded_file($_FILES["fichier"]["tmp_name"][$cle], $destination);
                $i++;
            }
        }
        echo "$i fichiers chargés";
    }
    // affichage du formulaire
    else
    {
        ?>
        <p><form method="POST" enctype="multipart/form-data" action="multific.php">
        <?php
            // affichage des 5 zones de fichiers
            for ($i=0;$i<maxfic;$i++)
            {
                echo "Entrer le nom de fichier numéro $i :";
            }
            ?>

            <input type="file" name="fichier[]" size="30"><br>
            <?php
            // fin des zones de fichiers
            }
            ?>

            <input type="submit" value="Envoyer" name="B1">

        </form>
        <?php
        //fin du else pour le formulaire
    }
    ?>
</body>
```

Remarque nous avons utilisé ici la fonction DEFINE pour définir une constante du script, la syntaxe de cette fonction est :

DEFINE(*nom de la constante*(chaîne de caractères), *valeur*)

Quelques fonctions utiles

PHP fournit un grand nombre de fonctions, nous allons ici présenter celles qui nous semblent les plus utiles.

La fonction include

La fonction ***include*** permet d'inclure dans un script PHP un autre script sauvegardé dans un fichier php. La syntaxe de la fonction est :

include("nom du fichier à inclure");

Par exemple, dans la mesure où la plupart des scripts vont faire appel à la connexion à la base de données, on peut conserver les instructions de connexion dans un fichier nommé connect.php et l'inclure dans les autres scripts. Le fichier connect.php, contiendra sera :

```
<?
mysql_connect("host","utilisateur","mot de passe") or die(mysql_error());
mysql_select_db("mabase") or die(mysql_error());
?>
```

Les autres fichiers du site contiendront alors dans le script PHP, l'instruction

```
include("connect.php");
```

Cette fonction peut aussi être utilisée, par exemple, pour avoir une présentation standard des différentes pages du site, sous forme de tableau, une structure générale pouvant être :

```
<table width="100%">
<tr>
<td colspan="2">
<? //en tête standard après fusion de deux cellules
include("entete.php");
?></td>
</tr>
<tr>
<td width="100">
<? //menu à gauche largeur de 100 pixels
include("menu.php");
?>
</td>
<td>
<? //lecontenu de la page
include("page_contenu.php");
?>
</td>
</tr>
<tr>
<td colspan="2">
<? // bas de page standard après fusion des deux cellules
include("bas_page.php");
?>
</td>
</tr>
</table>
```

Ce qui donnerait une présentation telle que :

| Site CDthèque | | | |
|--------------------------------|--|------|---------|
| Sélection | Pour consulter notre catalogue, veuillez indiquer vos critères de choix ci-dessous : | | |
| | Genre musical : | Tous | |
| | Compositeur : | Tous | |
| | Interprète : | Tous | Envoyer |
| Votre panier | | | |
| Accueil | | | |
| Site réalisé par les étudiants | | | |

Fonctions chaînes de caractères

Les fonctions `strlen`, `strtolower`, `strtoupper`

La fonction `strlen` renvoie le nombre de caractères d'une chaîne, la syntaxe est :

```
$longueur= strlen("chaîne");
```

La fonction `strtolower` transforme une chaîne de caractères en la chaîne équivalente écrite en minuscule, la syntaxe est :

```
$chaîne= strtolower("chaîne");
```

La fonction `strtoupper` transforme une chaîne de caractères en la chaîne équivalente écrite en majuscule, la syntaxe est :

```
$chaîne= strtoupper("chaîne");
```

La fonction `str_replace`

Cette fonction permet de remplacer des chaînes de caractères dans un texte, la syntaxe de cette fonction est :

```
$chaîne= str_replace(cible, remplacement, texte);
```

Les trois éléments peuvent être soit des chaînes de caractères soit des tableaux.

Si `cible` et `remplacement` sont deux chaînes de caractères toutes les occurrences de `cible` dans `texte` (ou tous les éléments du tableau `texte`) sont remplacés par `remplacement`.

Si `cible` seul est un tableau, chaque occurrence d'un des éléments de `cible` est remplacé par `remplacement` dans `texte`.

Si `cible` et `remplacement` sont des tableaux, le nombre d'éléments de `remplacement` (pris en compte) est inférieur ou égal au nombre d'éléments de `cible`. Chaque occurrence du premier élément de `cible` est remplacé dans `texte` par le premier élément de `remplacement`, puis le second etc.. Si le nombre d'éléments de `remplacement` est inférieur au nombre d'éléments de `cible`, les éléments surnuméraires de `cibles` sont remplacés par une chaîne vide.

Les fonctions `explode` et `implode`

La fonction ***explode*** permet de séparer une chaîne de caractères en fonction d'un séparateur, chacun des éléments séparés est stocké dans un tableau, à partir de l'indice 0. La syntaxe de la fonction est :

```
$tableau= explode("séparateur", "chaîne")
```

Par exemple en utilisant comme séparateur l'@, on pourra isoler le nom du domaine dans une adresse mel :

```
$tableau= explode("@", "gil.dupond@ici.com");
```

donnera comme résultat :

\$tableau[0]="gil.dupond" et \$tableau[1]=ici.com

La fonction ***implode*** est la duale de la fonction ***explode***, elle permet de recoller les éléments d'un tableau en une chaîne de caractères, chaque élément étant séparé du suivant par le séparateur choisi. La syntaxe de la fonction est :

\$chaine=***implode***("séparateur","tableau");

Par exemple, on pourra construire une adresse mel, à partir du nom et du domaine avec le séparateur "@". Avec le tableau précédent :

\$adresse=***implode***("@",\$tableau);

donnera :

\$adresse=gil.dupond@ici.com

La fonction ***strtok***

La fonction ***strtok*** est une extension de la fonction ***explode*** vue précédemment dans la mesure où l'on peut utiliser plusieurs séparateurs, la décomposition de la chaîne de caractères se fait alors en jetons, c'est à dire en éléments se trouvant entre deux séparateurs (identiques ou non). Contrairement à la fonction ***explode***, qui extrait tous les éléments dans un tableau, la fonction ***strtok*** extrait au fur et à mesure les jetons, et doit donc être appelée plusieurs fois pour extraire tous les jetons, la syntaxe de la fonction est différente selon que l'on extrait le premier jeton ou les suivants. Pour extraire le premier jeton, on utilisera la syntaxe :

\$mot=***strtok***("phrase","liste des séparateurs");

où liste des séparateurs est la suite des caractères séparateurs (par exemple " ." pour l'espace, la virgule ou le point).

Dans les appels suivants, pour extraire les autres jetons, on utilisera la syntaxe :

\$mot=***strtok***("liste des séparateurs");

A la fin de la chaîne de caractères à analyser, la fonction retourne la valeur booléenne FALSE, qu'il faudra tester avec == ou !=, pour éviter d'arrêter avec un jeton vide (par exemple deux espaces consécutifs dans l'exemple précédent).

Exemple : comptage des mots d'un texte :

<?

```
// Le texte à analyser a été entré dans une zone de texte
$texte=$_POST["texte"];
// les séparateurs : espaces, virgule, point, point virgule, parenthèse, deux points
$separateur=" ,.;():";
// extraction du premier token
$mot=strtok($texte,$separateur);
while ($mot!==FALSE)
{
    // si le token n'est pas vide
    if($mot)
    {
        // initialisation du compteur ou incrémentation(le mot est mis en minuscule)
        $mot=strtolower($mot);
        if (isset($compteur[$mot]) )
        {
            $compteur[$mot]++;
        }
        else
        {
            $compteur[$mot]=1;
        }
    }
    // on passe au token suivant
    $mot=strtok($separateur);
}
//Ecriture des résultats
foreach($compteur as $token=>$nb)
echo "$token : $nb<br>";
?>
```

Les fonctions `preg_match` et `preg_match_all`

Les fonctions `preg_match` et `preg_match_all` permettent de vérifier l'existence et d'extraire une ou toutes les chaînes de caractères d'un texte suivant un modèle. Nous allons tout d'abord voir la définition d'un modèle (expressions régulières) puis l'utilisation de ces fonctions avec le modèle ainsi défini.

Les expressions régulières

Ce paragraphe n'est qu'une introduction aux expressions régulières, et non exhaustif. Pour en savoir plus vous pouvez consulter le manuel PERL :

(<http://www.perl.com/doc/manual/html/pod/perlre.html>)

Les expressions régulières sont des modèles de recherche précisément définis, qui viennent du monde d'Unix et du langage PERL en particulier. Ces expressions peuvent à première vue paraître ésotériques mais leur usage s'avère souvent essentiel. A l'aide de ces modèles de recherche, il est possible, par exemple, de parcourir le contenu de variables et d'en extraire des contenus déterminés. Ainsi, à l'aide d'expressions régulières il est possible d'extraire d'un fichier les noms des fichiers images utilisés.

Structure d'une expression régulière

Une expression régulière est un modèle de chaîne de caractère, ce modèle doit être entouré de séparateurs, on utilise généralement le `/`. Entre ces séparateurs est décrit le type de chaîne que l'on recherche, un certain nombre de conventions permettent de générer des modèles très généraux. Le séparateur de droite peut-être suivi d'indicateurs pour spécifier le type de recherche que l'on veut effectuer, nous ne considérerons ici que deux de ces indicateurs :

- `i` : pour indiquer que l'on ne prend pas en considération la casse (majuscule, minuscule)
- `U` : pour indiquer que l'on stoppe la recherche à la chaîne de caractères la plus courte correspondant au modèle (gourmandise limitée), sinon la recherche trouvera la chaîne la plus longue possible.

Commençons par un exemple, recherchons les numéros de téléphone français sous format national dans un texte. La chaîne de caractères que nous avons à retrouver est de la forme suivante :

0 suivi d'un des chiffres (1,2,3,4,5,6 ou 8) puis de 8 chiffres.

Le modèle correspondant sera :

```
/0[1-68]\d{8}/
```

Explication : les `/` sont les séparateurs, le premier caractère du numéro doit être 0, ensuite nous devons avoir soit un chiffre entre 1 et 6 soit 8, c'est ce qui est exprimé par le crochet, enfin le numéro doit se terminer par 8 chiffres : le `\d` indique un chiffre (équivalent à `[0-9]`) et le nombre entre accolades indique le nombre de répétition de ce type de caractères.

Supposons maintenant que l'on admette que l'utilisateur puisse ou non séparer son numéro par tranche de deux chiffres soit par un espace soit par un tiret soit par un point. Dans ce cas il nous faudra introduire ce nouvel élément dans notre modèle en précisant son aspect facultatif par le symbole `?` :

```
/0[1-68][- .?(\d{2}[- . ]?){3}\d{2}/
```

Le début du modèle est identique, nous avons ensuite indiqué entre crochet les 3 caractères séparateurs possible, ce crochet est suivi d'un point d'interrogation signifiant que ce caractère n'est pas obligatoire, ensuite nous devons avoir trois groupes de deux chiffres suivi ou non du caractère séparateur, pour indiquer la répétition par 3 de cet ensemble, nous l'avons mis entre parenthèses, et enfin deux chiffres.

Remarque : sur ces exemples nous avons vu que certains caractères : ?, [, {, \., } avait une signification particulière dans le modèle, si nous voulions les retrouver dans un modèle il faudrait les faire précéder du caractère d'échappement \, nous reviendrons sur ce cas plus loin.

Expressions régulières pour signes distincts

Dans une chaîne de caractères, il est possible

- rechercher un caractère déterminé
- rechercher plusieurs caractères déterminés
- rechercher un caractère d'un passage déterminé

Ces expressions ne sont pas suffisantes par elles mêmes mais doivent être combinées comme dans les exemples précédents.

| Expression | Interprétation |
|--------------|---|
| a | contient un 'a' |
| [ab] | contient un 'a' ou un 'b' |
| [A-Z] | contient une majuscule (sans accent ni cédille) |
| [0-9] | contient un chiffre |
| \d | contient un chiffre - exactement |
| \D | contient un caractère qui n'est pas un chiffre |
| [-\d] | contient un chiffre ou le caractère moins |
| [\[] | contient un crochet |
| [a-zA-Z0-9_] | contient un caractère du type lettre (sans accent ni cédille), de type chiffre ou tiret de soulignement |
| \w | contient un caractère des types lettre, chiffre ou tiret de soulignement - (presqu'exactly comme le précédent; si ce n'est que les caractères avec accent peuvent être reconnus suivant la configuration du système |
| \W | contient un caractère qui n'est ni une lettre, ni un chiffre, ni un tiret de soulignement les caractères avec accent peuvent être exclus, dépend de la configuration du système |
| [^a-zA-Z] | contient un caractère qui n'est pas une lettre (sans accent) |

Quand le signe recherché est un signe réservé à l'intérieur d'expressions régulières, comme par exemple un +,., vous devez le faire précéder de la barre oblique inversée (\+). Les signes réservés sont : +?.*^\$()[]{}|\.

Expressions régulières pour chaînes de caractères

Ces expressions permettent de spécifier :

- une chaîne de caractères déterminée
- une chaîne de caractères avec des opérateurs de remplacement (jokers)
- une chaîne de caractères au début ou à la fin d'un mot
- une chaîne de caractères au début ou à la fin d'une ligne

Ce genre d'expressions régulières intervient dans la construction d'un modèle pour la recherche de l'occurrence d'un certain mot, d'une chaîne de caractères partielle.

Les expressions régulières sont ainsi constituées de mots reliés par des règles grammaticales, la liaison se faisant par des opérateurs. La forte compression du langage le rend à prime abord assez hermétique.

| Expression régulière | Effet |
|----------------------|---|
| aus | la recherche doit contenir 'aus' –comme dans 'pause' ou 'Gauss' |
| aus? | la recherche doit contenir soit 'aus' soit 'au' |
| a. | la recherche doit contenir une chaîne de deux caractères |

| | |
|---------------|---|
| | commençant par 'a' |
| a+ | la recherche doit contenir au moins une fois la lettre 'a' |
| a* | la recherche doit contenir pas de ou un nombre quelconque de lettres 'a' |
| te.s | la recherche doit contenir une chaîne de 4 lettres commençant par te et finissant par s. |
| te.+s | la recherche doit contenir une chaîne d'au moins 4 caractères commençant par te et finissant par s |
| te.?s | la recherche doit contenir une chaîne d'au moins 3 caractères commençant par te et finissant par s |
| x{10,20} | la recherche doit contenir entre 10 et 20 'x' à la suite |
| x{10,} | la recherche doit contenir au moins 10 'x' à la suite |
| x.{2}y | la recherche doit contenir une chaîne de 4 caractères commençant par x et finissant par y |
| Jean\b | la recherche doit contenir 'Jean' mais pas 'Jeannot' (limite du mot) |
| \bvers | la recherche doit contenir 'verser' ou 'versatile' mais pas 'envers' (limite du mot) |
| \bvers\b | la recherche doit contenir 'vers' mais pas 'envers' ni 'verser' (limite du mot) |
| \bvers\B | la recherche doit contenir 'verser' mais ni 'vers' et ni 'envers' (limite du mot et limite du mot "négative") |
| ^Jean | la recherche doit contenir 'Jean' seulement au début du passage à parcourir |
| Jean\$ | la recherche doit contenir 'Jean' seulement à la fin du passage à parcourir |
| (jpg gif bmp) | la recherche doit contenir 'jpg' ou gif ou bmp |
| \$Nom | interprète le contenu de la variable \$Nom comme expression |

Utilisation et utilité des parenthèses dans un modèle.

Les parenthèses sont utilisées dans les expressions régulières à deux fins, exceptées bien sûr dans le cas où elles sont précédées de la barre oblique inversée auquel cas l'expression régulière considère le caractère (ou) comme faisant partie de la recherche.

Tout d'abord les parenthèses peuvent être utilisées comme une mise en facteur d'éléments, c'est ce que nous avons vu sur l'exemple du numéro de téléphone ci-dessus ou dans le cas de recherche disjonctive.

L'utilité des parenthèses est aussi une aide à la récupération de chaînes dans la mesure où un niveau de variables (indice d'un tableau) sera associé au même niveau de parenthèse. Par exemple, le modèle

/code postal : (\d{5})/

permettra de récupérer dans la variable de niveau 0, toute la chaîne "code postal : 75001", tandis que la variable de niveau 1 ne récupérera que le code postal seul :75001.

Exemple modèle pour retrouver les fichiers images dans une fichier HTML

Nous ne nous intéresserons ici qu'à la balise image du fichier (pas au image de fond de page par exemple)le fichier image est défini après la propriété scr, le nom du fichier est entre guillemets et se termine par jpg, gif ou png. Le modèle sera alors le suivant :

/src="(\\w[.]*\\. (jpg|gif|png))"/Ui

Explications : on spécifie que le nom de fichier doit commencer par un caractère autorisé (lettre, chiffre ou soulignement) puis un nombre quelconque de caractères, suivi du point

(noter la barre oblique inversée) et de l'extension. La "gourmandise" est limitée à la première occurrence de ce type de chaîne pour ne pas dépasser le nom du fichier par exemple si deux images sont sur la même ligne. Il est de plus spécifié que l'on ne tient pas compte de la casse. Il est laissé au lecteur la modification de ce modèle pour prendre en compte d'éventuels espaces entre le signe = et le mot clé scr ou le premier guillemet.

Enfin le nom de fichier est conservé dans la variable de niveau 1, la variable de niveau 0 contenant l'ensemble de la chaîne trouvée.

Les fonctions ***preg_match*** et ***preg_match_all***

Ces deux fonctions permettent de rechercher dans un texte et éventuellement de stocker dans un tableau multidimensionnel les résultats de la recherche d'une chaîne de caractères suivant un modèle.

Ces fonctions retournent le nombre d'occurrences, correspondant au modèle, trouvées dans le texte. La syntaxe de ces fonctions est :

```
$noc=preg_match_all($modele,$texte,$result);  
$nb=preg_match($modele,$texte,$result);
```

où *\$modele* correspond à la chaîne de caractère servant de modèle, *\$texte* est la chaîne à explorer et *\$result* est un tableau contenant les chaînes de la variable *\$texte* satisfaisant le modèle.

Dans le cas de ***preg_match*** la recherche s'arrête à la première correspondance, la valeur retournée est donc 0 ou 1, dans le cas de ***preg_match_all*** la recherche se poursuit jusqu'à la fin du texte. La fonction ***preg_match*** est donc surtout utilisée pour vérifier l'adéquation à un modèle, tandis que ***preg_match_all*** sera plutôt utilisé pour extraire des informations du texte.

Exemple 1 : vérification d'un numéro de téléphone au format standard. Le script suivant permet de vérifier qu'un numéro entré par l'utilisateur peut correspondre à un numéro de téléphone :

```
<body>  
<?php  
//On vérifie que l'utilisateur a cliqué sur le bouton de validation  
if(isset($_POST["valid"]))  
{  
    $modele="/0[1-68][- ]?(\\d{2}[- ]?){3}\\d{2}/";  
    $nb=preg_match ($modele,$_POST["Tel"],$result);  
    if($nb)  
    {  
        //Le numéro est valide  
        echo "Nous avons bien noté que votre Tel est :".$result[0][0];  
        exit();  
    }  
    //Sinon affichage d'un message et demande à nouveau du numéro  
    else echo "<b> Votre numéro est mal entré, veuillez recommencer</b>";  
}  
?>  
<form method="POST" action="tel.php">  
    <p>N°Tel <input type="text" name="Tel" size="26">  
    <input type="submit" value="OK" name="valid"></p>  
</form>  
</body>
```

Le formulaire est affiché dans deux cas : avant toute entrée et en cas de non conformité du numéro entré.

Le modèle utilisé est celui qui a été présenté en introduction, si le numéro est valide on l'affiche, remarquez les deux indices du tableau **\$result**, le premier correspond au tableau du premier niveau de parenthèses, le second est l'indice (la clé) du seul élément de ce tableau.

Exemple 2 : extraction des noms des fichiers images d'une page HTML. Nous utiliserons ici le modèle défini précédemment, les noms de fichiers seront stockés dans le tableau d'indice 1 correspondant au niveau de parenthèse du nom de fichier. La fonction utilisée sera ***preg_match_all***, puisque nous voulons tous les noms de fichiers, on éliminera les doublons avec la fonction ***array_unique*** (voir ci dessous). Le script est le suivant :

```
<?php
if($f=@fopen("index.htm","r"))
{
    // lecture du fichier htm
    $text=fread($f,filesize("index.htm"));
    fclose($f);
}
else
{
    //Erreur à l'ouverture
    echo "Impossible d'ouvrir le fichier";
    exit;
}
$modelle='/src="([\w].*\.(jpg|gif))"/Ui';
$nb=preg_match_all ($modelle, $text, $images);
//Suppression des doublons dans les noms de fichier (indice 1 du tableau resultat)
$nomfic=array_unique($images[1]);
echo "$nb correspondances et".count($nomfic)." fichiers distincts<br>";
//Affichage des noms des différents fichiers
foreach($nomfic as $value)
echo "$value <br>";
?>
```

Les noms de fichiers se trouvent dans le tableau \$result[1], le tableau \$result[0] contenant l'ensemble de la chaîne de caractères src="..."

Avec comme fichier *index.htm* :

```
<body>
<p><font size="7">Le
département SIAD</font></p>
<p>Les membres du département :</p>
<table border="1" width="100%">
  <tr>
    <td align="center">
      </td>
    <td align="center">
      </td>
    <td align="center">
      </td>
    <td align="center">
      </td>
  </tr>
  <tr>
    <td align="center">
      </td>
    <td align="center">
      </td>
    <td align="center">
      </td>
    <td align="center">
      </td>
  </tr>
  <tr>
    <td align="center">
      </td>
```

```

</tr>
</table>
</body>

```

Le script donne alors le résultat suivant :

```

11 correspondances et 10 fichiers distincts
images/logohec.gif
images/YP.jpg
images/MT.jpg
images/MHD.jpg
images/PLM.jpg
images/BLM.jpg
images/HT.jpg
images/JO.jpg
images/GM.jpg
images/JMG.jpg

```

Fonctions sur les tableaux

Nous allons rapidement présenter ici les fonctions les plus utiles sur les tableaux associatifs, en particulier les fonctions de tri.

Les fonction min, max, count

PHP fournit les fonctions usuelles pour déterminer le minimum et le maximum d'un tableau unidimensionnel (vecteur). La syntaxe est la suivante :

```

$elt=min($montablo);
$elt=max($montablo);

```

Il est possible de remplacer la variable \$montablo, par une liste de variables.

Dans la mesure où les clés des tableaux ne sont pas nécessairement des entiers, et où il est possible avec la fonction **unset** de supprimer des éléments d'un tableau, PHP fournit une fonction retournant le nombre d'éléments d'un tableau. Le résultat de cette fonction est un entier et la syntaxe est la suivante :

```

$nbelt=count($montablo);

```

La fonction array_unique

La fonction **array_unique** permet d'éliminer les doublons dans un tableau, la fonction renvoie un tableau après suppression des doublons, en préservant les clés, en cas de doublon la première clé est conservée. La syntaxe est la suivante :

```

$tablo=array_unique($montablo);

```

Par exemple le script suivant donnera comme résultat un tableau à 3 éléments dont les clés seront 0,1,4 et les éléments 1,2,7 :

```

<?php
$montab=array(1,2,1,2,7,7,1);
$ntab=array_unique($montab);
print_r($ntab);
?>

```

le résultat du script est :

```

Array ( [0] => 1 [1] => 2 [4] => 7 )

```

Remarque : si l'on veut retrouver les clés "naturelles" numériques consécutives (0,1,...), on utilisera la fonction `array_values`, qui remplace les clés par des entiers consécutifs, le script suivant :

```
<?php
$montab=array(1,2,1,2,7,7,1);
$ntab=array_unique($montab);
$ntab=array_values($ntab);
print_r($ntab);
?>
```

conduira à la sortie :

Array ([0] => 1 [1] => 2 [2] => 7)

Les fonctions de tri

PHP fournit différentes fonctions pour trier un tableau unidimensionnel (vecteur), en ordre croissant ou décroissant, sur les valeurs avec ou sans conservation des indices ou sur les clés. Ces fonctions de tri ont toutes deux arguments, le tableau à trier et le deuxième argument facultatif est le type de comparaison utilisé pour le tri :

| Valeur | Type de comparaison |
|------------------------------------|--|
| <code>SORT_NUMERIC</code> | Compare des nombres |
| <code>SORT_REGULAR</code> (défaut) | Compare les types mixtes comme des chaînes et les numériques comme des nombres |
| <code>SORT_STRING</code> | Compare tout comme des chaînes |

Les fonctions de tri sont des tris "sur place", donc modifie le tableau initial, qui est perdu après l'opération de tri, à moins d'avoir été sauvegardé dans une autre variable au préalable.

Tri sur les valeurs sans conservation des clés

Les fonctions de tri de base sont ***sort*** (tri dans l'ordre croissant) et ***rsort*** (tri dans l'ordre décroissant) qui ordonnent les valeurs mais détruisent les clés éventuelles. La syntaxe de ces fonctions est :

sort(\$tabo,type de comparaison);

ou

rsort(\$tabo,type de comparaison);

Le script suivant utilise la fonction ***sort*** :

```
<?php
$voca=array(
    "homme"=>"man",
    "femme"=>"woman",
    "enfant"=>"child",
    "garçon"=>"boy",
    "fille"=>"girl"
);
//tri avec destruction des clés
sort($voca);
print_r($voca);
?>
```

Le résultat de ce programme sera alors :

Array ([0] => boy [1] => child [2] => girl [3] => man [4] => woman)

Tri sur les valeurs avec conservation des clés

Comme il est souvent utile de conserver les clés, PHP possède deux fonctions de tri sur les valeurs avec conservation de clés, ce sont les fonction ***asort*** (tri croissant) et ***arsort*** (tri décroissant), dont la syntaxe est identique à celle de `sort` :

asort(\$tabo,type de comparaison);

ou
arsort(\$tabo,type de comparaison);

Le script suivant utilise la fonction **arsort** :

```
<?php
$voca=array(
    "homme"=>"man",
    "femme"=>"woman",
    "enfant"=>"child",
    "garçon"=>"boy",
    "fille"=>"girl"
);
//tri avec conservation des clés
arsort($voca);
foreach($voca as $cle=>$valeur)
    echo "La valeur : $valeur pour la clé : $cle<br>";
?>
```

La sortie correspondante sera :

La valeur : boy pour la clé : garçon
La valeur : child pour la clé : enfant
La valeur : girl pour la clé : fille
La valeur : man pour la clé : homme
La valeur : woman pour la clé : femme

Tri sur les clés

Il est aussi possible de trier un tableau par rapport à ses clés, pour cela on utilise la fonction **ksort** pour un tri croissant ou la fonction **krsort** pour un tri décroissant. La syntaxe de ces fonction est identique à celle de sort :

arsort(\$tabo,type de comparaison);
ou
arsort(\$tabo,type de comparaison);

Le script suivant utilise la fonction **krsort** :

```
<?php
$voca=array(
    "homme"=>"man",
    "femme"=>"woman",
    "enfant"=>"child",
    "garçon"=>"boy",
    "fille"=>"girl"
);
//tri décroissant sur les clés
krsort($voca);
foreach($voca as $cle=>$valeur)
    echo "A la clé $cle est associée la valeur $valeur<br>";
?>
```

et la sortie correspondante :

A la clé homme est associée la valeur man
A la clé garçon est associée la valeur boy
A la clé fille est associée la valeur girl
A la clé femme est associée la valeur woman
A la clé enfant est associée la valeur child

Tri sur plusieurs valeurs

Quand on utilise un tableau à deux dimensions, c'est à dire un tableau de vecteurs, il est possible de faire un tri multicritère, c'est à dire que l'on trie d'abord sur un vecteur et en cas

d'ex æquo, sur un deuxième vecteur, et ainsi de suite. En fait la fonction de PHP, n'impose pas que les vecteurs soient dans un même tableau, mais simplement que les vecteurs aient le même nombre de composants. La syntaxe de la fonction est la suivante :

array_multisort(vecteur1,sens1,vecteur2,sens2,...);

où sens1 est une constante indiquant si le tri sur le vecteur le précédant est en ordre croissant ou décroissant, cette constante est définie par PHP :

| | |
|-----------|-------------|
| SORT_ASC | croissant |
| SORT_DESC | décroissant |

Exemple, on veut trier la liste des employés par département, puis par nom à l'intérieur des départements. Le tableau suivant donne cette liste :

| Nom | Prénom | Département |
|--------|---------|-------------|
| Leroy | Emile | Finance |
| Moi | Gilles | Compta |
| Lui | Alain | Finance |
| Dupont | Mathieu | Compta |
| Dupont | Gérard | Compta |

Le script suivant trie ce tableau en ordre croissant sur les départements, décroissant sur les noms et croissant sur les prénoms, puis affiche le résultat dans un tableau :

```
<?php
$lesgens=array(
    "Nom"=>array("Leroy","Moi","Lui","Dupont","Dupont"),
    "Prenom"=>array("Emile","Gilles","Alain","Mathieu","Gérard"),
    "Dpt"=>array("Finance","Compta","Finance","Compta","Compta")
);
array_multisort(    $lesgens["Dpt"],SORT_ASC,
                   $lesgens["Nom"],SORT_DESC,
                   $lesgens["Prenom"],SORT_ASC);

?>
<table border=1>
<tr><td align="center">Nom</td>
<td align="center">Prénom</td>
<td align="center">Département</td></tr>
<?php
foreach($lesgens["Nom"] as $clé=>$valeur)
{
echo "<tr>";
    foreach($lesgens as $pclé=>$stab)
        echo "<td>".$lesgens[$pclé][$clé]."</td>";
echo "</tr>";
}
?>
</table>
```

La sortie obtenue est :

| Nom | Prénom | Département |
|--------|---------|-------------|
| Moi | Gilles | Compta |
| Dupont | Gérard | Compta |
| Dupont | Mathieu | Compta |
| Lui | Alain | Finance |
| Leroy | Emile | Finance |

Remarques :

1. il serait plus "naturel" que les sous tableaux correspondent aux caractéristiques d'un individu c'est à dire que la déclaration du tableau \$lesgens soit plutôt :

```
$lesgens=array(  
    array("Nom"=>"Leroy","Prénom"=>"Emile","Département"=>"Finance"),  
    array("Nom"=>"Moi","Prénom"=>"Gilles","Département"=>"Compta"),  
    array("Nom"=>"Lui","Prénom"=>"Alain","Département"=>"Finance"),  
    array("Nom"=>"Dupont","Prénom"=>"Mathieu","Département"=>"Compta"),  
    array("Nom"=>"Dupont","Gérard"=>"Mathieu","Département"=>"Compta"),  
);
```

mais dans ce cas nous ne pourrions pas utiliser la fonction ***array_multisort***.

2. Il est indispensable de mettre comme paramètres de la fonction tous les sous tableaux sinon les tableaux absents ne seraient pas alignés sur le tri. En particulier, il est impossible en cas d'ex æquo, de ne pas trier sur un des vecteurs restants.

Nous verrons plus loin comment traiter ce problème.

Définition de fonctions par le programmeur

Comme tous les langages informatiques, PHP permet au programmeur de définir ses propres fonctions. Une fonction retourne généralement une valeur, dans le cas où il n'y a pas de retour la fonction joue le rôle de ce qui dans d'autre langage (comme le pascal) s'appelle une procédure.

Syntaxe

La syntaxe est la suivante :

```
function nom_fonc(param1,param2,...)
{
    corps de la fonction
    return (valeur_à_retourner);
}
```

L'appel de la fonction se faisant par une instruction du type :

```
monresult=nom_fonc(p1,p2,..)
```

La fonction suivante permet de retrouver toutes les clés correspondant à la valeur maximale d'un tableau :

```
function clemax($tablo)
{
    $max=max($tablo);
    foreach ($tablo as $cle=>$valeur)
        if ($valeur==$max)
            $mcle[]=$cle;
    return($mcle);
}
```

L'exécution par le programme suivant :

```
$mont=array(1,5,2,4,5,1,5,2);
print_r(clemax($mont));
```

donnera :

Array ([0] => 1 [1] => 4 [2] => 6)

Les clés 1,2 et 6 étant les clés correspondant au maximum (5) du tableau.

Portée des variables

Les variables utilisées dans une fonction sont locales, les variables utilisées dans le script ne sont donc pas directement accessibles dans la fonction. Par exemple le script suivant :

```
<?php
$a=1; //variable globale du script
function demo()
{
    echo 'Valeur de $a en entrée : ', $a, "<br />";
    $a=2;
    echo 'Valeur de $a modifiée avant la sortie : ', $a, "<br />";
}
demo();
echo 'Après retour de la fonction $a vaut : ', $a, "<br />";
?>
```

donnera :

Valeur de \$a en entrée :

Notice: Undefined variable: a in **Essai-cours/portee.php** on line **5**

Valeur de \$a modifiée avant la sortie : 2
Après retour de la fonction \$a vaut : 1

Pour prendre en compte la variable \$a du script, il faut la déclarer comme globale dans la fonction, le script modifié :

```
<?php
$a=1; //variable globale du script
function demo()
{
    Global $a ; // permet d'accéder à la variable du script
    echo 'Valeur de $a en entrée : ', $a, "<br />";
    $a=2;
    echo 'Valeur de $a modifiée avant la sortie : ', $a, "<br />";
}
demo();
echo 'Après retour de la fonction $a vaut : ', $a, "<br />";
?>
```

donnera :
Valeur de \$a en entrée : 1
Valeur de \$a modifiée avant la sortie : 2
Après retour de la fonction \$a vaut : 2

Récurtivité

PHP supporte la récursivité ce qui permet d'écrire certaines fonctions de façon plus élégante. On appelle récursive toute fonction qui de façon directe ou indirecte fait appel à elle-même. Exemple : la fonction puissance nième (pour $n \geq 0$) d'un nombre x peut être définie par : $x^0 = 1$ et $x^n = x * x^{n-1}$

Ce qui se traduit en PHP par la fonction suivante :

```
function puissance($n,$x)
{
    if($n>0)
    { return ($x*puissance($n-1,$x)) ;}
    else {return(1) ;}
}
```

Remarque importante :

La procédure est arrêtée par le test conditionnel *else (1)*. Dans toute fonction récursive, il doit y avoir une condition d'arrêt, qui empêche toute boucle "infinie" (ici il serait bon de vérifier que n est positif ou nul).

Exécution d'une procédure ou fonction récursive :

Nous allons suivre ici l'exécution de l'instruction de l'appel $y = puissance(3, 5)$;

Première phase : création de la pile

1. Une première "copie" de la fonction est créée en mémoire avec des variables locales \$x=5 et \$n=3. L'exécution est suspendue à l'instruction (**return(\$x*)**) car puissance(2,5) est inconnue
2. Une deuxième copie de la fonction est alors créée avec les variables locales \$x=5 et \$n=2. L'exécution est suspendue à l'instruction (**return(\$x*)**)

3. Une troisième copie de la fonction est créée avec les variables locales $\$x=5$ et $\$n=1$
4. L'exécution est suspendue à l'instruction (***return(\$x*)***)
5. Cette fois ci, avec $\$x=5$ et $\$n=0$, la copie s'exécute et retourne $\text{puissance}(0,5)=1$

Deuxième phase : dépilage

6. La troisième copie peut se terminer et retourne alors $\text{puissance}(1,5)=5$
7. La deuxième copie peut se terminer et retourne alors $\text{puissance}(2,5) = 25$
8. La première copie se termine et retourne à l'instruction d'appel $\text{puissance}(3,5)=125$

La récursivité permet de programmer de façon plus simple et souvent plus « lisible », mais en revanche cette méthode utilise plus de mémoire que les méthodes classiques et le temps d'exécution d'un programme peut rapidement dégénérer : par exemple pour le calcul des nombres de Fibonacci (définis par $\text{fibo}(1)=\text{fibo}(2)=1$ et $\text{fibo}(n)=\text{fibo}(n-1)+\text{fibo}(n-2)$ pour $n \geq 3$), on pourra comparer les deux programmes suivants :

```
function fiborecur($n)
{
    if ($n<3)
    {return (1) ;}
    else
    {return (fiborecur($n-1)+fiborecur($n-2)) ;}
}
```

et

```
function fiboboucle($n)
{
    $x=1 ; //initialisation pour $n=1
    $y=1 ; //initialisation pour $n=2
    for($i=3 ; $i<=$n ; $i++)
    {
        $a=$x ; //tampon pour ne pas écraser $x ;
        $x=$y ;
        $y=$a+$y ;
    }
    return($y) ;
}
```

le script d'appel suivant permet de comparer les temps de calcul en secondes :

```
$d1=time();
echo fiborecur(34),"<br />";
echo "Temps d'exécution récursif :",time()-$d1,"s<br />";
$d1=time();
echo "<br />", fiboboucle(34),"<br />";
echo "Temps d'exécution non récursif :",time()-$d1,"s<br />";
```

Retour sur les fonctions de tri

Nous avons vu plus haut que PHP ne possède pas réellement de fonction permettant de trier un tableau sur plusieurs clés. Ce problème peut être résolu en utilisant la fonction `usort` qui permet de trier un tableau en utilisant une fonction de comparaison définie par le programmeur.

Fonction de comparaison

Une relation d'ordre total étant définie sur des éléments, une fonction de comparaison est une fonction qui admet comme paramètre deux éléments **a** et **b** et qui retourne les valeurs 1, 0 ou -

1 suivant que **a** est plus grand, égal ou plus petit que **b**. Par exemple considérons la relation d'ordre définie sur les chaînes de caractère par :

- $S_1 \prec S_2$ si le nombre de caractères de S_1 est inférieur au nombre de caractères de S_2 .

Par exemple "Grand" est inférieur à "Petite", mais égal à "Petit".

La fonction de comparaison associée à cette relation d'ordre s'écrit :

```
function compar_nb_car($a,$b)
{
    $na=strlen($a);
    $nb=strlen($b);
    if($na>$nb) {return(1);}
    elseif($na<$nb) {return(-1);}
    else {return(0);}
}
```

Utilisation de usort

La syntaxe de usort est : usort(\$tab,"nom_fonction") où le premier paramètre est un array et le second la chaîne de caractères donnant le nom de la fonction de comparaison. Nous allons illustrer cette utilisation avec quelques exemples.

Avec la fonction précédente

Le programme suivant

```
include("comp_nb_car.php");// fichier contenant la fonction
$montab=array("Maurice","Jean-Paul","Leopoldine", "Jules","Albert", "Louis",
              "Laurence","Jean","Gilles","Robert","Théodore","Francis");
usort($montab,"compar_nb_car");
//appel de la fonction avec la fonction de comparaison
for($i=0;$i<count($montab);$i++)
{
    echo $montab[$i],"<br />";
}
```

donnera comme résultat :

Jean
Jules
Louis
Robert
Gilles
Albert
Francis
Maurice
Laurence
Théodore
Jean-Paul
Leopoldine

Fonction de comparaison modifiée

Nous allons traiter spécialement le cas d'égalité, en rangeant alphabétiquement les noms de même longueur :

```

function compar_nb_car2($a,$b)
{
    $na=strlen($a);
    $nb=strlen($b);
    if($na>$nb) {return(1);}
    elseif($na<$nb) {return(-1);}
    else {
        // traitement de l'égalité
        If($a==$b){return(0);}
        Else{return( ($a < $b) ? -1 : 1;)}
    }
}

```

Le programme appelant

```

include("comp_nb_car2.php");//fichier contenant la fonction
$montab=array("Maurice","Jean-Paul","Leopoldine", "Jules","Albert", "Louis",
    "Laurence","Jean","Gilles","Robert","Théodore","Francis");
usort($montab,"compar_nb_car2");
//appel de la fonction avec la fonction de comparaison
for($i=0;$i<count($montab);$i++)
{
    echo $montab[$i],"<br />";
}

```

donnera alors :

Jean
 Jules
 Louis
 Albert
 Gilles
 Robert
 Francis
 Maurice
 Laurence
 Théodore
 Jean-Paul
 Leopoldine

Tri multicritère d'un tableau

Les clés critères du tri sont stockés dans un tableau noté \$cles et le sens du tri dans un autre tableau \$sens, ces deux tableaux sont globaux. La fonction de comparaison (compmul) appelle une fonction récursive (compare) : en cas d'égalité seulement on fait appel au critère suivant.

```

function compare($tab1,$tab2,$ind)
{
    global $cles;
    global $sens;
    $la=$cles[$ind];
    $dir=$sens[$ind];
    $indmax=count($cles)-1;

```

```

        if ($tab1[$la]<$tab2[$la])
        {return(-$dir);}
        elseif($tab1[$la]>$tab2[$la])
        {return($dir);}
        Else //cas d'égalité sur le critère
        {
            if($ind<$indmax) //s'il reste d'autres critères
            {return(compare($tab1,$tab2,$ind+1));}
            else return(0);
        }
    }
}
function compmul($tab1,$tab2)
{
    return(compare($tab1,$tab2,0));
}

```

Le script suivant permet de tester cette fonction :

```

<?php
include("comparemul.php");
$exemple= array(
array("Département"=>"Finance","Nom"=>"Dupont","Prénom"=>"Jules","Age"=>20),
array("Département"=>"Compta","Nom"=>"Machin","Prénom"=>"Gilles","Age"=>59),
array("Département"=>"Finance","Nom"=>"Dupont","Prénom"=>"Alain","Age"=>40),
array("Département"=>"Market","Nom"=>"Dupont","Prénom"=>"Claude","Age"=>30),
array("Département"=>"Compta","Nom"=>"Lui","Prénom"=>"Jacques","Age"=>28),
array("Département"=>"Finance","Nom"=>"Dupont","Prénom"=>"Pierre","Age"=>27)
);
$cles=array("Département","Nom","Age"); // les critères de tri
$sens=array(1,-1,1); // le sens ascendant=1, descendant=-1
usort($exemple,"compmul");
?>

<table border=1>
<<?php
$titre=array_keys($exemple[0]);
echo "<tr>";
    foreach($titre as $valeur)
        echo "<td>".$valeur."</td>";
    echo "</tr>";
for($i=0;$i<count($exemple);$i++)
{
    echo "<tr>";
    foreach($exemple[$i] as $valeur)
        echo "<td>".$valeur."</td>";
    echo "</tr>";
}

?>
</table>

```

L'exécution donnera le résultat suivant :

| Département | Nom | Prénom | Age |
|-------------|--------|---------|-----|
| Compta | Machin | Gilles | 59 |
| Compta | Lui | Jacques | 28 |
| Finance | Dupont | Jules | 20 |
| Finance | Dupont | Pierre | 27 |
| Finance | Dupont | Alain | 40 |
| Market | Dupont | Claude | 30 |