
Web et bases de données

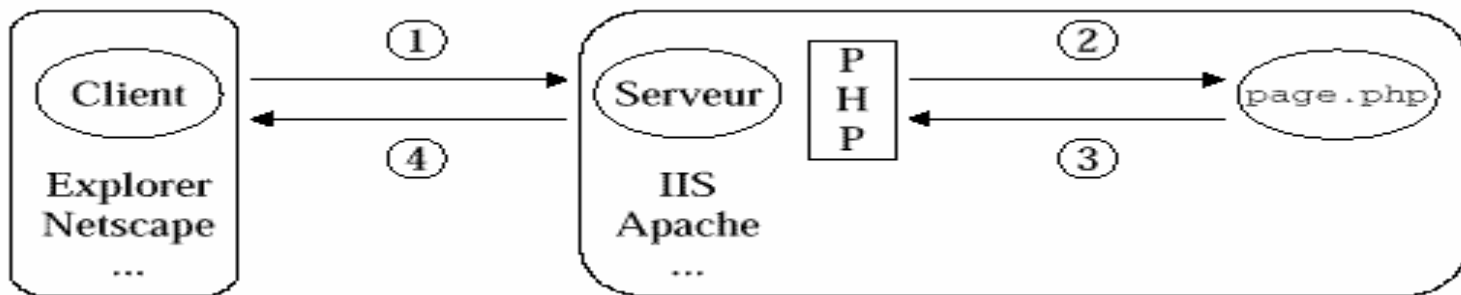
Présentation de PHP

Sadok Ben Yahia, PhD
Faculté des Sciences de Tunis
Sadok.benyahia@fst.rnu.tn
<http://www.cck.rnu.tn/sbenyahia>



PHP : Introduction

- ☒ PHP : Langage de script permettant d'insérer de la programmation dans des pages web dynamiques
- ☒ Conçu en 1994 par Rasmus Lerdorf, première version publique en 1995, puis PHP/FI (1995), PHP3 (1997), PHP4 (2000)
- ☒ PHP ressemble beaucoup à C/C++ et supporte d'un grand nombre de bases de données (Gratuit, fonctionne sous UNIX et Windows)
- ☒ Programme s'exécutant côté serveur Web : du code embarqué dans une page HTML entre les balises <? et ?>
- ☒ extension .php pour les pages PHP (stockés sur le serveur (comme des docs)
- ☒ Sont désignés par une URL `http:// www. lip2. tn/ page.Php` et le chargement de l'URL provoque l' exécution côté serveur



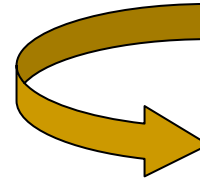
PHP : Illustration du fonctionnement

```
<HTML> <BODY>
<HTML> <BODY>
<H1> Table des factorielles</
H1>
<?
for ( $i= 1,$ fact= 1 ; $i< 4 ; $i++, $
fact*=$ i )
{ echo "$i! = $fact <BR>"; }
?>
</ BODY> </ HTML>
```

résultat =
HTML
généré via la
du code PHP

code PHP

Invocation



Exécution
côté serveur



ce qui est renvoyé au client

```
<HTML> <BODY>
<H1> Table des
factorielles</ H1>
1! = 1 <BR>
2! = 2 <BR>
3! = 6 <BR>
</ BODY> </ HTML>
```

Un programme PHP, c'est...

- Un fichier HTML...
- dans lequel on trouve des balises PHP :

```
<html>
<head><title>TEST</title></head>
<body>
  <p>
    Il est <?echo date("H:i");?>.
  </p>
</body>
</html>
```

Dans la pratique... (surtout pour les gros projets)

- Un programme PHP est un ensemble d'instructions PHP qui affichent :
 - du code HTML
 - ou autre chose :
 - ascii
 - PostScript, PDF
 - GIF, JPEG, PNG, ...
- L'approche « pages » est désormais supplantée par l'approche « composants »

Une page HTML avec des instructions PHP

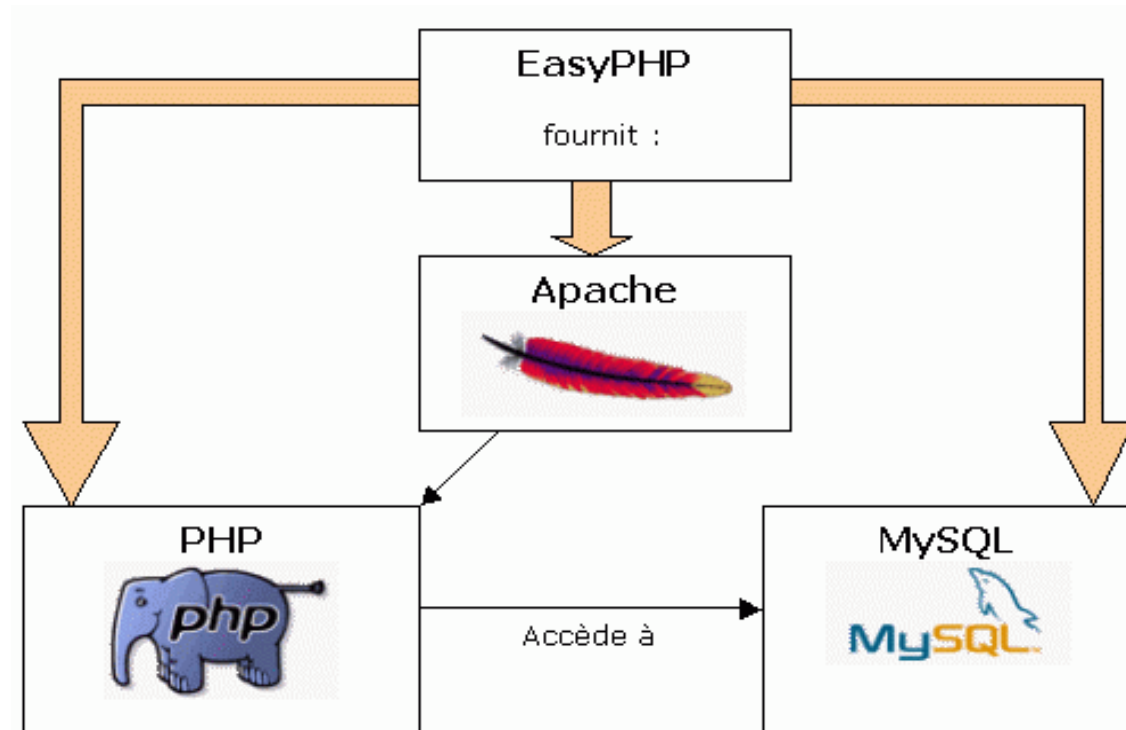
```
<html>                                <?php
  <head>                                codes php ...
    <title>                              ?>
      Bonjour !                          Ou
    </title>                              <?
  </head>                                codes php ...
  <body>                                  ?>
    <p>
      sur le serveur, il est exactement
      <? echo date("H:i:s") ; ?>
    </p>
  </body>
</html>
```

Un programme PHP

```
<?
```

```
class sortie {  
    function sortie($titre) // constructeur  
        { $this->titre = $titre ; }  
    function debut()  
        { echo "<html><head>$this->titre</head><body>" ;  
        }  
    function fin()  
        { echo "</body></html>" ; }  
}  
  
$s = new sortie("Bonjour !") ;  
$s->debut() ;  
echo "<p>sur le serveur, il est exactement "  
    .date("H:i:s")  
    . "</p>" ;  
$s->fin() ;
```

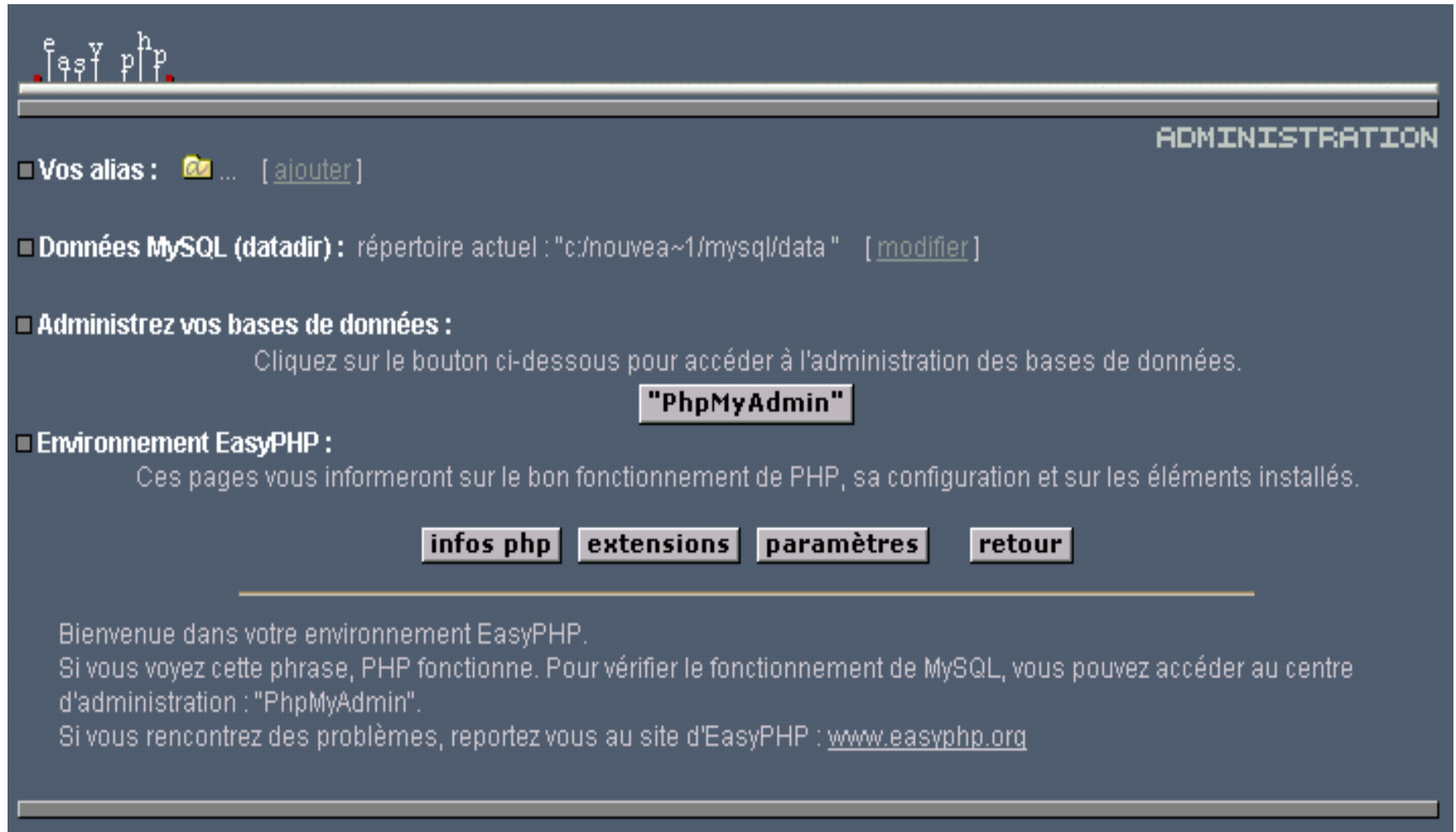
L'outil EasyPHP




L'outil EasyPHP

- Les fonctions proposés par EasyPHP :
- Arrêter et Redémarrer les serveurs Apache et MySQL.
- Accéder au "Web local", c'est-à-dire la racine des sites webs.
- Un panneau d'administration en PHP
- Un outil de configuration d'EasyPHP
- L'accès aux logs
- L'aide

L'administration EasyPHP



ADMINISTRATION

- ▣ Vos alias :  ... [ajouter]
- ▣ Données MySQL (datadir) : répertoire actuel : "c:/nouvea~1/mysql/data" [modifier]
- ▣ Administrez vos bases de données :
Cliquez sur le bouton ci-dessous pour accéder à l'administration des bases de données.
"PhpMyAdmin"
- ▣ Environnement EasyPHP :
Ces pages vous informeront sur le bon fonctionnement de PHP, sa configuration et sur les éléments installés.
infos php **extensions** **paramètres** **retour**

Bienvenue dans votre environnement EasyPHP.
Si vous voyez cette phrase, PHP fonctionne. Pour vérifier le fonctionnement de MySQL, vous pouvez accéder au centre d'administration : "PhpMyAdmin".
Si vous rencontrez des problèmes, reportez vous au site d'EasyPHP : www.easyphp.org

Le langage PHP

Bases

Variables / Constantes

Expressions / Opérateurs

Structures de contrôle

Fonctions / Inclusions de fichiers

Syntaxe de base

- Séparateur d'instructions : point-virgule

```
<? instruction1 ;  
    instruction2 ?>
```

- Commentaires

- à la C :

```
/* commentaire */
```

- à la C++ :

```
// fin de ligne commentée
```

- à la shell :

```
# fin de ligne commentée
```

Entiers et réels

■ Entiers

1234	# <i>décimal</i>
-123	# <i>négatif</i>
0123	# <i>octal (=83)</i>
0x12	# <i>hétéradécimal (=18)</i>

■ Flottants (réels)

1.234	
1.2e3	# <i>(=1200)</i>

Chaînes de caractères

- Entre guillemets :
 - variables remplacées
 - `\n`, `\r`, `\t`, `\\`, `\$`, `\"`, `\167`, `\x5f`
- Entre apostrophes :
 - variables non remplacées
 - `\\`, `\'`
- Indices de 0 à l-1 (`$str[0]` à `$str[$l-1]`)

PHP: Syntaxe ... Chaînes de caractères

- entre guillemets (les ' ' marchent aussi) **`$couleur=" rouge";`**
- substitutions de variables à l'intérieur d'une chaîne
`$figure=" carré $couleur foncé";` ⇒ `"carré rouge foncé"`
- encodages des caractères spéciaux **`\$ || \n \t`**
- entre apostrophes : sans substitution, ni encodage
`$figure= 'carré $couleur foncé'; "carré $couleur foncé"`
- longueur d'une chaîne **`strlen($ figure)`**
- comparaison avec l'opérateur == **`$figure == $couleur`**
- nombreuses fonctions de manipulation disponibles
- Variables dynamiques var. dont l'identificateur est la valeur d'une variable

`ex. : $var=" hello"; $$ var=" world";`

`echo "$var $hello ${$ var}"; ! "hello world world"`

Conversion de chaînes

- On prend le premier mot de la chaîne
 - si mot numérique [-]n[.n][en], la valeur
 - sinon 0

- Exemples :

1 + "10.5"	11.5
1 + "-1.3e3"	-1299
1 + "bob-1.3e3"	1
1 + "bob3"	1
1 + "10 cochons"	11
"10.0 cochons" + 1	11
"10.0 cochons" + 1.0	11.0

Tableaux

- **Scalaire ou associatif**

`$tab[12], $tab["foo"], $tab[$n]`

- **Multi-dimensionnels**

`$coord["top"][$x][$y]`

- **Création :**

`array()`, `list()`, assignation directe

- **Accès :**

`count()`, `each()`, `next()`, `prev()`

- **Tri :**

`asort()`, `arsort()`, `ksort()`, `usort()`, ...

Tableaux (création)

- Directe

- `$tab[0] = "bonjour" ;`
- `$t["café"] = 2 ;`

- Avec la fonction `array()`

- `$dejeuner = array(
 "entrée" => "salade",
 "plat" => "choucroute",
 "boisson" => "eau",
 "dessert" => "flan",
 "café" => TRUE) ;`

Typage

- Le type d'une expression est déterminé par le contexte dans lequel elle est utilisée
- Il peut être forcé
`settype()`
- Type casting traditionnel
`$entier = (integer) 1.1 // (=1)`
- Il n'existe pas de type booléen dédié
 - 0 est faux,
 - non nul est vrai

Variables

- `Nom = $(lettre|_)(lettre|chiffre|_)*`

```
$vingtaine = 20 ; // ok
$20n = 20 ; // erreur
$_20n = 20 ; // ok
```

- Case-sensitive

- Références (~alias, ~pointeur)

```
$foo = "bonjour" ;
$bar = &$foo ;
$bar = "au revoir" ;
echo $foo ; // au revoir
```

Variables prédéfinies

- Accessibles par `phpinfo()`
- Apache
- Environnement
 - selon l'O.S.
 - attention ! . remplacés par des _
- PHP (dont paramètres CGI)
 - `HTTP_GET_VARS`, `HTTP_POST_VARS`
`HTTP_COOKIE_VARS`, `HTTP_POST_FILES`, ...

[EasyPHP] - Administration - Microsoft Internet Explorer

Fichier Edition Affichage Favoris Outils ?

Précédente Recherche Favoris Média

Adresse <http://127.0.0.1/home/index.php?to=phpinfo> OK Liens

- Données MySQL (datadir) : répertoire actuel : "C:\Program Files\EasyPHP\mysql\data" [modifier]
- Administrez vos bases de données : Cliquez sur le bouton ci-dessous pour accéder à l'administration des bases de données. **"PhpMyAdmin"**
- Environnement EasyPHP : Ces pages vous informeront sur le bon fonctionnement de PHP, sa configuration et sur les éléments installés.
 - [infos php](#)
 - [extensions](#)
 - [paramètres](#)
 - [retour](#)

PHP Version 4.2.0



System	Windows NT 5.1 build 2600
Build Date	Apr 20 2002 18:36:03
Server API	Apache
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\WINDOWS\php.ini
Debug Build	no
Thread Safety	enabled

This program makes use of the Zend Scripting Language Engine:
 Zend Engine v1.2.0, Copyright (c) 1998-2002 Zend Technologies



Fichiers Réseau

Internet

démarrer formationPHP et + Microsoft PowerPoint ... [EasyPHP] - Administr... FR 11:12

Portée des variables

- Par défaut, une variable est locale
- Pour accéder à une variable globale depuis une fonction :
 - `global $a ;`
 - `$GLOBALS["a"]`

Portée des variables (exemple)

```
$a = "bonjour" ;
```

```
$b = "hello" ;
```

```
function f($x)
```

```
{
```

```
    global $b ;
```

```
    echo $a ;           // rien
```

```
    echo $b ;           // hello
```

```
    echo $GLOBALS["a"] ; //
```

```
    bonjour
```

```
}
```


Variables de variables (exemple)

```
$a = "bonjour" ;
```

```
$$a = "monsieur" ;
```

```
echo "$a ${$a}" ; // bonjour monsieur
```

```
echo "$a $bonjour" ; // idem
```

Propriétés des variables

■ Existence

- `isset()`

■ Type

- `gettype()`

- `is_long()`, `is_double()`, `is_string()`

- `is_array()`, `is_object()`

■ Contenu

- `empty()`

Constantes

- Variables à assignation unique
 - `define("MA_CONSTANTE", 45) ;`
- Quelques constantes prédéfinies
 - `__FILE__`, `__LINE__`, `PHP_VERSION`, `TRUE`, `FALSE`, ...

```
function
affiche_erreur($fichier,$ligne,$message)
{
    echo "<p>$fichier, ligne $ligne :
$message</p>" ;
}

affiche_erreur(__FILE__,__LINE__,"aie !") ;
```

Expressions

- En PHP, tout est expression (comme en C)
- Conséquences :
 - on peut trouver 100 décimales de π en une seule ligne de code
 - on peut facilement perdre en lisibilité ;-)

```
int a=10000,b,c=8400,d,e,f[8401],g;
main()
{
    for(;b-c;) f[b++]=a/5;
    for(;d=0,g=c*2;c -= 14,printf("%.4d",e+d/a),e=d%a)
        for (b=c;d += f[b]*a,f[b]=d% --g,d /=g--, --b; d *=b) ;
}
```

Opérateurs(1)

■ Arithmétiques

- +, -, *, / (quotient), % (reste)

■ Binaires (bit à bit)

- & (et), | (ou), ^ (ou excl.), ~ (non), << et >> (décalages)

■ Comparaison

- == (égal), === (identique), !=, >, <, <=, >=

Opérateurs(2)

- Incrémentation (++), décrémentation (--)

```
$a = 5 ;
```

```
$b = 2 ;
```

```
$c = ( 2 * --$a ) % $b-- ;
```

- Logiques

```
if ( ($a && !$b) || $c )
```

```
...
```

- Caractères

- (composition)

Opérateurs(3)

- Exécution

```
$sortie = `ls -al` ;  
echo "<pre>$sortie</pre>" ;
```

- Assignations composées

`+=, -=, *=, /=, %=, .=, &=, |=, ^=, ~=, <<=, >>=`

- Alternative

`expr_bool ? res_si_vrai : res_si_faux`

Opérateurs (priorité)

- Par ordre de priorité décroissante

```
,  
or  
xor  
and  
print  
= += -= *= /=  
. = %=  
&= |= ^= ~=  
<<= >>=  
? :  
||  
&&  
|  
  
^  
&  
== != ===  
<<= >>=  
<< >>  
+ - .  
* / %  
! ~++ --  
(type) @  
[  
new
```


Structures de contrôle

- Conditionnelle
- Alternative / alternative multiple
- Choix multiple
- Boucles
 - « tant que »
 - « répéter »
 - « pour »
 - « pour chaque »

Alternative (multiple)

```
if (condition1)
{
    /* . . . */
}
elseif (condition2)
{
    /* . . . */
}
elseif . . .
. . .
else
{
    /* . . . */
}
```

PHP :Exemples de structure

```
if ($a>$b) echo 'A > B';  
if ($a>$b)  
{  
echo 'A > B';  
$b = $a;  
}  
if ($a>$b)  
{ echo 'A > B';  
}  
else  
{  
echo 'A <= B';  
}
```

```
switch($i)  
{  
case 0: echo 'i=0'; break;  
case 1: echo 'i=1'; break;  
case 2: echo 'i=2'; break;  
}  
  
switch($ch) {  
case "a": echo 'A'; $ch="A";  
break;  
case "b": echo 'B'; $ch="B";  
break;  
case "c": echo 'C'; $ch="C";  
break;  
default: echo 'Ni "a" ni "b" ni  
"c";  
}
```

Tant que / répéter

```
while (condition)
{
    /* . . . */
}
```

```
do
{
    /* . . . */
}
while (condition) ;
```

Boucle « pour »

```
for ( expr1 ; expr2 ; expr3 )  
{  
    /* . . . */  
}
```

- est équivalent à

```
expr1 ;  
while ( expr2 )  
{  
    /* . . . */  
    expr 3 ;  
}
```

PHP :Exemples de structure

□ **while(expr) {instr}**

```
$i=1;
```

```
while ($i<=10) { echo $i++; }
```

□ **do {instr} while(expr)**

```
$i=1;
```

```
do
```

```
{ echo $i++; }
```

```
while ($i<=10);
```

□ **for (expr1; expr2; expr3) {instr}**

```
for ($i=1; $i<=10; $i++) { echo $i ;}
```

Boucle « pour chaque » (1)

```
foreach ($liste as $valeur)
{
    instructions;
}
```

- est équivalent à :

```
foreach ($tableau as $cle=>$valeur)
{
    instructions;
}
```

Boucle « pour chaque » (2): Exemple

```
<html> <body BGCOLOR=#FFFFFF>
<B><U>La boucle foreach</U><Br><Br><B>
<?
# J'affecte les éléments à ma liste
$liste = array( "4 bougies", "cagouille", "php4");
# J'affiche le nombre d'éléments de ma liste ainsi que son contenu
$i=0;
print "Etat de la liste <U>avant</U> le tri<BR>";
foreach($liste as $valeur)
{
print "\$liste[$i] vaut <FONT COLOR=green>$valeur</FONT><BR>";
$i++;
}
print "<HR SIZE=8>";
# Essai sur un tableau associatif
$tab = array( "Ali"=>"Alia", "dom"=>"tom");
print "Le tableau associatif tab contient";
print " <FONT COLOR=green>".count($tab)."</FONT> éléments<BR>";
foreach ($tab as $cle=>$valeur)
{
print "\$tab[<FONT COLOR=green>$cle</FONT>] vaut";
print " <FONT COLOR=green>$tab[$cle]</FONT><BR>";
}
?>
</body></html>
```


Choix multiple

```
switch (expr)
{
  case val1:
    /* . . . */
    break ;
  case val2:
    /* . . . */
    break ;
  . . .
  default:
    /* . . . */
}
```

Ruptures de séquence

■ **continue**

- ❑ arrêter l'itération courante
- ❑ passer à la suivante

■ **break [n]**

- ❑ arrêter l'itération courante
- ❑ sortir de la boucle courante
(ou des n boucles imbriquées)

Fonctions (exemple)

```
function exemple ($arg_1,  
                 $arg_2,  
                 ...  
                 $arg_n)  
{  
    echo "dans exemple<br>\n" ;  
    return $val_ret ;  
}
```

- Notes :
 - peut retourner une valeur
 - peut être appelée sans affectation (comme en C)

Fonctions : valeur par défaut des arguments

```
function augmente($valeur,$ajout=1)
{
    return $valeur + $ajout ;
}
```

```
$x = 5 ;
```

```
echo augmente($x,4) . "<br>" ;           // 9
echo augmente($x) . "<br>" ;           // 6
```

- argument vide \neq argument absent

Fonctions : passage des arguments par référence

```
$a = $b = 0 ;           // initialisation de $a et $b

function f($p1,&$p2)    // déclaration de la
fonction f
{
    $p1 = $p2 = 5 ;
}

f($a,$b) ;             // appel de la fonction f

// affichage des valeurs de $a et $b
echo  "\$a=$a, \$b=$b<br>\n" ;
```

Inclusion de fichiers

- **require** ("fichier.inc") ;
 - inclut le fichier fichier.inc
 - évaluation en pre-processing

- **include** ("fichier.inc") ;
 - inclut le fichier fichier.inc
 - évaluation à chaque fois

- Inclusion unique
 - **include_once** ()
 - **require_once** ()

Gestion de fichiers

- Avec PHP, la création ou la lecture de fichiers est assez simple. Il existe une **multitude de fonctions dédiées à l'utilisation des fichiers**.
- La communication entre le script PHP et le fichier est repérée par une **variable**, indiquant l'état du fichier et que l'on peut passer en paramètre aux fonctions spécialisées pour le manipuler.

Gestion de fichiers: Ouverture

- La fonction de base est la fonction **fopen()**. : permet d'ouvrir un fichier, que ce soit pour le lire, le créer, ou y écrire.
entier fopen(chaine nomdufichier, chaine mode);
- Le **mode** indique le type d'opération qu'il sera possible d'effectuer sur le fichier après ouverture. Il s'agit d'une lettre (une chaîne de caractères)
 - **r** (comme *read*) indique une ouverture en lecture seulement
 - **w** (comme *write*) indique une ouverture en écriture seulement (la fonction crée le fichier s'il n'existe pas)
 - **a** (comme *append*) indique une ouverture en écriture seulement avec ajout du contenu à la fin du fichier (la fonction crée le fichier s'il n'existe pas)
- Lorsque le mode est suivie du caractère **+** celui-ci peut être lu et écrit.
- Faire suivre le mode par la lettre **b** entre crochets indique que le fichier est traité de façon binaire.

Gestion de fichiers: Ouverture

```
$fp = fopen("../fichier.txt", "r"); //lecture
```

```
//écriture depuis début du fichier
```

```
$fp = fopen("ftp://phpTunis.com/pub/fichier.txt", "w");
```

```
//écriture depuis fin du fichier
```

```
$fp = fopen("http://igalaxie.com/fichier.txt", "a");
```

- De plus, la fonction `fopen` permet d'ouvrir des fichiers présents sur le web grâce à leur URL. Voici un script permettant de récupérer le contenu d'une page d'un site web:

```
<?  
$fp = fopen("http://www.commentcamarche.net", "r"); //lecture du fichier  
while (!feof($fp)) { //on parcourt toutes les lignes  
    $page .= fgets($fp, 4096); // lecture du contenu de la ligne  
}  
?>
```

Gestion de fichiers: Ouverture

- Il est généralement utile de tester si l'ouverture de fichier s'est bien déroulée ainsi que d'éventuellement stopper le script PHP si cela n'est pas le cas:

```
<?  
if (!$fp = fopen("http://www.commentcamarche.net", "r")) {  
echo "Echec de l'ouverture du fichier";  
exit;  
}  
else {  
// votre code;  
}  
?>
```

- Un fichier ouvert avec la fonction `fopen()` doit être fermé, à la fin de son utilisation, par la fonction `fclose()` en lui passant en paramètre l'entier retourné par la fonction `fopen()`

Gestion de fichiers: Lecture et écriture

- Lire son contenu (après ouverture) et d'y écrire des informations grâce aux fonctions:
- **fputs()** (aussi parfois appelée **fwrite()**, les deux noms sont équivalents, on parle d'alias) permettant d'écrire une chaîne de caractères dans le fichier :
boolean fputs(entier Etat_du_fichier, chaine Sortie);
renvoie 0 en cas d'échec, 1 dans le cas contraire
- **fgets()** permettant de récupérer une ligne du fichier
chaîne fgets(entier Etat_du_fichier, entier Longueur);
 - ❑ Le paramètre *Longueur* désigne le nombre de caractères maximum que la fonction est censée récupérer sur la ligne.
 - ❑ renvoie 0 en cas d'échec, 1 dans le cas contraire
 - ❑ Etant donné que la fonction *fgets()* récupère à chaque appel une nouvelle ligne du fichier, il est essentiel, pour récupérer l'intégralité du contenu d'un fichier de l'insérer dans une boucle *while*.
- La fonction **feof()**, teste si la fin du fichier n'a pas été atteinte, en tant que test de la boucle *while*.

Gestion de fichiers: Lecture et écriture

<?

```
if (!$fp = fopen("fichier.txt","r")) {  
    echo "Echec de l'ouverture du fichier";  
    exit;  
}  
else {  
    while(!feof($fp)) {  
        // On récupère une ligne  
        $Ligne = fgets($fp,255);  
        // On affiche la ligne  
        echo $Ligne;  
        // On stocke l'ensemble des lignes dans une variable  
        $Fichier .= $Ligne;  
    }  
    fclose($fp); // On ferme le fichier  
}  
?>
```

Gestion de fichiers: Lecture et écriture

- Pour stocker des infos dans le fichier, il faut dans un premier temps ouvrir le fichier en écriture en le créant si il n'existe pas. On a donc le choix entre le mode 'w' et le mode 'a'.
- Préférer le second puisque le pointeur se trouve en fin de fichier (autrement dit on écrit à la suite de ce qui se trouve dans le fichier au lieu d'écraser le contenu existant éventuellement déjà).

```
<?  
$fp = fopen("php_8_fichier.txt","a"); // ouverture du fichier en  
écriture  
fputs($fp, "\n"); // on va a la ligne  
fputs($fp, "$nom/$email"); // on écrit le nom et email dans le fichier  
fclose($fp);  
>
```

Gestion de fichiers: Les fonctions utiles

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

boolean is_dir(chaine Nom_du_fichier);

renvoie 1 si il s'agit d'un répertoire, 0 dans le cas contraire

```
<?
```

```
if (!is_dir("install"))
```

```
{
```

```
echo "Il ne s'agit pas d'un répertoire";
```

```
}
```

```
else {
```

```
echo "Il s'agit bien d'un répertoire";
```

```
}
```

```
?>
```

Gestion de fichiers: Les fonctions

- ***is_executable()*** permet de savoir si le fichier dont le nom est passé en paramètre est exécutable
 - *boolean is_executable(chaine Nom_du_fichier);*
renvoie 1 si le fichier est exécutable, 0 dans le cas contraire
- ***is_file()*** permet de savoir si le fichier dont le nom est passé en paramètre ne correspond ni à un répertoire, ni à un lien symbolique
 - *boolean is_file(chaine Nom_du_fichier);*
renvoie 1 si il s'agit d'un fichier, 0 dans le cas contraire
- ***is_link()*** permet de savoir si le fichier dont le nom est passé en paramètre correspond à un lien symbolique
 - *boolean is_link(chaine Nom_du_fichier);*
renvoie 1 si il s'agit d'un lien symbolique, 0 dans le cas contraire