Campus numérique
francophone de Tunis

Formation TRANSFER

Tunis, du 14 au 18 Avril 2003

Atelier
Web dynamique
PhP + MySQL

AYARI Mejdi
mejdi.ayari@auf-francophonie.org

Qu'est-ce que PhP

Présentation :

PHP est un langage de script qui s'inclut dans le langage HTML. La syntaxe du langage PHP provient du C, du Java et du Perl, avec un petit nombre de fonctions inédites par rapport à ces langages. Le but du langage PHP est de permettre aux développeurs de site web d'écrire rapidement des pages web dynamiques.

Ce qui distingue le PHP des langages de script comme le JavaScript est que le code est exécuté sur le serveur et non par le navigateur. Le client ne reçoit que le résultat, sans aucun moyen d'avoir accès au code qui a produit l'affichage.

Apport de PhP :

Le langage PHP possède les mêmes fonctionnalités que les autres langages permettant d'écrire des scripts CGI, comme par exemple:

- Collecter des données.
- Générer dynamiquement des pages web
- Envoyer et recevoir des cookies.

La plus grande qualité du langage PHP est le support d'un grand nombre de bases de données. Réaliser une page web dynamique interfaçant une base de données devient extrêmement simple. PHP en supporte un grand nombre dont : Adabas D, dBase, Empress, FilePro, Informix, InterBase, mSQL, MySQL, Oracle, PostgreSQL, Solid, Sybase, Velocis, Unix dbm.

Le langage PHP inclut par ailleurs le support des services utilisant les protocoles tels que IMAP, SNMP, NNTP, POP3 ou encore HTTP.

Evolution de PhP :

PHP est né avec le site **de Rasmus Lerdof** en 1994: une page personnelle, avec son CV qui permettait àl'origine de conserver une trace des utilisateurs. A l'époque **PHP supportait déjà des requêtes SQL** et, comme cela arrive souvent sur le Web, des internautes ont rapidement voulu leur propre copie du programme. Rasmus a donc décidé de mettre en ligne la version 1.0 de PHP (**Personal Home Page**). A la surprise de son auteur, PHP est devenu rapidement très populaire. Conséquence : une multitude de suggestions ont atterri dans sa boîte aux lettres. Certes, PHP 1.0 pouvait contrôler la saisie des utilisateurs et remplacer certaines commandes HTML. **Mais les utilisateurs commençaient à se demander si PHP n'était pas capable de mieux faire** : boucles, structures conditionnelles, etc. A l'instar de tous les autres langages de programmation modernes, la seconde étape logique de l'évolution de PHP consistait effectivement àadopter des fonctionnalités plus puissantes. Rasmus étudia donc la structure des autres langages, il se documenta àpropos de YACC et de Bison GNU pour mettre au point la version 2.0.

La version 2.0 permet au développeur d'intégrer des **instructions de programmation puissantes directement dans du code HTML**. Concrètement, un script PHP peut analyser les données soumises par un formulaire HTML, communiquer avec des bases de données et effectuer des calculs complexes àla volée. Et l'exécution de tels scripts est très rapide, car le **code source du langage est directement compilé dans le serveur Web Apache**. Ainsi, les **instructions PHP sont exécutées à l'intérieur même du serveur**, sans aucune ressource supplémentaire, contrairement aux CGI (Common Gateway Interface).

A partir de cette version 2 (appelé **PHP/FI**), PHP a représenté un **outil crédible pour le développement Web**, et plusieurs sites commerciaux n'ont pas hésité àl'adopter. En 1996, Clear INK mit en ligne SuperCuts et utilisa PHP pour **créer un système de navigation personnalisé**. En janvier 1999, plus de 100 000 sites avaient adopté le langage PHP.

Une communauté de développeurs s'est créée autour du langage. De nouvelles fonctions ont ainsi pu voir le jour et de nombreux bugs ont été rapidement résolus. **Zeev Suraski** et **Andi Gutmans** ont apporté des fonctions essentielles en écrivant un nouveau moteur d'analyse, celui de la version 2.0 générant de nombreuses erreurs. C'est pourquoi Rasmus décida de créer la version 3.0 de PHP et demanda àplusieurs programmeurs de le rejoindre dans ce travail. Outre Zeev et Andi, citons également **Stig Bakken**, **Shane Caraveo** et **Jim Winstead**.

Après six mois de développement, la version 3.0 de PHP sortit le 6 juin 1998. PHP3 est la troisième version de l'interprêteur de ce langage.
Actuellement, PhP est dans sa 4ème version.

La version 4.0 de PHP inclut de nombreuses nouvelles possibilités :

· **Le moteur Zend.**
Ce moteur (qui se charge du scripting du PHP) comporte de nouveaux mécanismes :

- Compteur d'accès.
- Gestion avancée des Objets.
- Implémentation d'un nouveau type booléen
- Extensibilité.

· **Le support des sessions HTTP.**
Le support natif des sessions HTTP était l'une des fonctionnalités plus demandées pour la sortie de la version 4.0 de PHP. Dans la version 3.0, ces sessions étaient gérer àtravers la PHPlib.

· **Les processus d'implémentation de modules sous UNIX.**
Il est maintenant possible de construire facilement et dynamiquement des modules PHP. Cette possibilité n'est pas encore incluse dans la version Bêta actuelle de PHP4.

· **Installation plus conviviale.**

Les informations de configuration du fichier php.ini peuvent maintenant être chargées via les fichiers de configuration Apache (pour les serveurs Apache) ou la base de registre (pour Windows 32 bits).

Syntaxe :

La syntaxe du PhP est très proche de celle du langage C. Une instruction est terminée par un point-virgule en fin de ligne.
on prendra soin d'insérer des commentaires, ce qui augmente la lisibilité. Il faut penser que dans un script PHP on peut trouver pêle-mêle du HTML, du script PHP ainsi que du JavaScript. On a donc tout interêt àse soucier de la lisibilité du code.

Il y a quatre méthodes pour insérer un script PhP dans une page Html :

- **< *?php*** echo (exemple en PhP) ***; ?>***
  (La méthode la plus utilisée)
- **< *?*** echo(exemple en PhP) ***; ?>***
- ***<SCRIPT LANGUAGE= 'PhP'>***echo (exemple en PhP) ***;</SCRIPT>***
  (Cette méthode devrait être familière àtout tilisateur de JavaScript ou de VBScript)
- ***<%*** echo(exemple en PhP) ; ***%>***
  (Utilisataion de la méthode d'ActiveServerPage)

Le serveur http comprend, dès la rencontre d'un de ces caractères d'échappement, qu'il doit commencer àtraiter le code comme un script PHP.
Il exécute le code PhP et envoie la sortie du script (ici générée par echo) vers la navigateur comme une partie du document.
Lorsqu'il a atteint la balise de fin du script PhP (par exemple ***?>***) le serveur Web revient en mode Html, et poursuit l'envoi du contenu du document au navigateur sans autre traitement côté serveur.

PhP supporte les commentaires comme en C et C++ :

- Pour commenter une seule ligne on utilise une double barre oblique (//) :
  Exemple :
  < ?php
  echo (exemple en php) ; //Ceci est un commentaire
  ?>
- Pour commenter plusieurs lignes on utilise (/*) au début du bloc du commentaire et (*/) àla fin.
  Exemple :
  < ?php
  /*Ceci est un commentaire
  qui s'étale
  sur plusieurs lignes
  */
  echo (exemple en php) ;
  ?>

Variables, constantes et types de données

Constantes :

Définition d'une constante :

Une constante est une valeur qui ne change pas : Par exemple la valeur de *pi* (environ 3,14).
Le recours à des constantes constitue un moyen pratique pour affecter une valeur à un identifiant, puis de faire référence àcet identifiant dans l'ensemble du programme.
Pour définir une constante, on fait appel àla fonction ***define()***.

Exemple 1 :
define(SOCIETE , Société internationale d'Import Export ***)*** ;
echo (Je travaille à la société : . SOCIETE) ;
Exemple 2 :
define(NL ,
\n ***);***

Le deuxième exemple consiste à définir une constante nommée NL représentant une balise d e saut de ligne Html suivi par un caractère de saut de ligne. C'est en pratique un raccourci de programmation, puisque la séquence <BR\ est fréquemment utilisée>.

 **Une constante, comme son nom le laisse entendre, n'est pas prévue pour être modifiée. La première définition sera celle qui servira à initialiser la constante. Les suivantes seront sans effet.**

Exemple :

```
<?php
define ("ZOPE", "Z Object Publishing Environment");
define ("ZOPE", "Z Object");
echo ZOPE;
// retournera toujours "Z Object Publishing Environment"
?>
```

 **La fonction *defined()* permet de vérifier si une constante existe ou non. Elle renvoie 1 si la constante existe, et 0 dans le cas contraire.**

Exemple :

```
if(defined(SOCIETE)) {
echo ( Je travaille à la société : . SOCIETE) ;
}
```

Constantes prédéfinies :

PhP possède plusieurs constantes prédéfinies. Parmi ces constantes on peut citer :

- ***TRUE*** = 1
- ***FAUX*** = 0 ou chaîne vide
- ***E_ERROR*** = Dénote une erreur autre qu'une "parsing error" (erreur d'analyse) qu'il n'est pas possible de corriger.
- ***E_WARNING*** =Dénote un contexte dans lequel le PHP trouve que quelque chose ne va pas. Mais l'exécution se poursuit tout de même. Ces alertes peuvent être récupérées par le script lui-même. Un exemple serait une expression régulière (regexp) invalide dans la fonction ereg().
- ***E_PARSE*** = L'analyseur a rencontré une forme syntaxique invalide dans le script. Correction de l'erreur impossible.

- ***E_NOTICE*** = Quelque chose s'est produite, qui peut être ou non une erreur. L'exécution continue. Par exemple, le cas de guillemets doubles (") non refermés, ou bien la tentative d'accéder àune variable qui n'est pas encore affectée.
- ***PHP_VERSION*** = La chaîne de caractères de présentation de la version du PHP qui est actuellement utilisée. Par exemple '4.0.0'.
- ***PHP_OS*** = Nom du système d'exploitation qui est utilisé par la machine qui fait tourner le PHP. Par exemple, 'Linux'.

Utilisation des variables :

En PHP, les variables sont représentées par un signe dollar suivi du nom de la variable. Une variable est automatiquement déclarée dès que vous lui affectez une valeur. L'affectation est réalisée àl'aide de l'opérateur (**=**).
Exemple :
$nom = "Mejdi";
$prix = 15,200 ;

Détermination de types :

La détermination de types permet de modifier explicitement le type de données d'une variable.

Exemple :
$var = 15,6 ; //$var est un double
$var = (int) $var ; //c'est maintenant un entier (valeur 15)
$var = (double) $var ; //c'est de nouveau un double (valeur 15,0)
$var_chaine = (string)$var ; //$var_chaine est une chaîne (valeur 15)

Les variables dynamiques :

Il est pratique d'avoir parfois des noms de variables qui sont variables. C'est-à-dire un nom de variable qui est affecté et utilisé dynamiquement. Une variable classique est affecté avec l'instruction suivante:
$a = "hello";
Une variable dynamique prend la valeur d'une variable et l'utilise comme nom d'une autre variable. Dans l'exemple ci-dessous, hello, peut être utilisé comme le nom d'une variable en utilisant le "$$" précédent la variable. c'est-à-dire :
$$a = "world";
A ce niveau, deux variables ont été définies et stockées dans l'arbre des symboles PHP: $a avec comme valeur "hello" et $hello avec comme valeur "world". Alors, l'instruction :
echo "$a $$a";
produira le même affichage que :
echo "$a $hello";
c'est-à-dire : hello world.

Fonctions utiles pour les variables :

PhP possède plusieurs fonctions internes permettant de travailler avec les variables :

- string ***gettype*** (mixed ***var***)
  Retourne le type de la variable PHP ***var***.
  Les chaînes de caractères que peut retourner la fonction sont les suivantes :
  - "integer"
  - "double"
  - "string"
  - "array"
  - "object"
  - class
  - "unknown type" (type inconnu)

Exemple :
```
if(gettype($saisie_utilisateur) ==    integer   ) {
      $age=$saisie_utilisateur ;
}
```

- int ***settype*** (string ***var***, string ***type***)
  Définit de façon explicite le type (***type***) de la variable ***var.***
  Le type peut être : integer, double, string, array ou object.
  La fonction settype renvoie TRUE en cas de succès, FALSE sinon.
- int ***isset*** (mixed ***var***)
  Sert àdéterminer si une variable possède une valeur.
  Renvoie TRUE si la variable ***var*** est définie, FALSE sinon.
  Exemple :
  ```
  $nom=   Ali    ;
  echo(isset($nom)) ; //TRUE
  ```
- int ***unset*** (mixed ***var***)
  Sert àdétruire une variable.
  Exemple :
  ```
  $nom =    Ali    ;
  unset($nom) ;
  if (isset($nom)) {
  echo(   Ceci NE s'affiche PAS   ) ;
  }
  ```

- int ***empty*** (mixed ***var***)
  Elle renvoie TRUE si la variable n'est pas définie, ou possède une valeur de 0 ou une chaîne vide. Sinon elle renvoie FALSE.

  Exemple :
  ```
  $var = 0;
  if (empty($var)) { //retourne TRUE
  print 'soit $var vaut 0, soit il n'est pas défini';
  }
  ```

PHP tient compte de la casse des caractères dans les noms de variables. Ainsi, $chaine ou $Chaine permet d'adresser deux variables bien distinctes. Notez enfin que PHP supporte les caractères accentués dans les noms de variable.

Quelles sont les valeurs par défaut de variables non initialisées ?

Cela dépend du type de variables. Les valeurs par défaut sont :

- **type numérique : 0.00**
- **type chaine : "" équivalent à NULL**
- **type booléen : false équivalent à 0**
- **type tableau : NULL avec une taille à 0**

Les tableaux :

Un tableau est composé d'un certain nombre d'éléments possédant chacune une valeur propore ainsi qu'une clé ou indice , permettant de faire référence àcet élément.
Par défaut, l'indice du premier élément d'un tableau est ***zéro***.

Initialiser un tableau :

Il y a plusieurs façons d'insérer des éléments dans un tableau. On peut tout simplement leur affecter une valeur, de la même façon que pour une autre variable :

Exemple :

```
$tableau[]="a";
$tableau[]="b";
$tableau[]=  c  ;
```

Ce code crée un tableau de trois éléments. Comme aucun indice n'a été spécifié de façon explicite entre les crochets [] , les éléments possèdent des indices respectifs par défaut 0, 1 et 2.

Le même code aurait pu être écrit en spécifiant des indices explicites :

Exemple :

```
$tableau[0]="a";
$tableau[1]="b";
$tableau[2]=  c  ;
```

Ce n'est cependant pas très pratique lorsque l'on souhaite insérer plusieurs valeurs d'un coup.

En pratique on utilisera plutôt la fonction **array()**. Il suffit pour cela de transmettre àarray() les valeurs devant être assignées au nouveau tableau :

Exemple :

```
$tableau = array(  a  ,  b  ,  c  ) ;
```

On peut utiliser l'opérateur **=>** afin de spécifier les indices particuliers des éléments d'un tableau.

Exemple :

```
$tableau = array(1=>"a",
                 2=>"b",
                 3=>"c") ;
```

Parcourir un tableau :

On peut procéder de plusieurs manières selon que le tableau est indexé en séquence ou que ses indices sont imprévisibles.

Tout tableau possède un pointeur interne. Ce curseur interne conserve la trace de l'élément actuellement actif.

Il est possible de déterminer la valeur de l'élément actif à l'aide de la fonction ***current()*** et l'indice de l'élément actif àl'aide de la fonction ***key()***

Exemple :

```
$pays[50] =  fr  ;
$pays[20] =  de  ;
$pays[10] =  us  ;
$pays[] =  ch  ;
```

$cle = key ($pays) ;

$valeur = current($pays) ;

echo(L'élément $cle est à$valeur) ;

Comme ce tableau vient d'être créé, l'élément actif est le premeir, si bien que ce code va imprimer : L'élément 50 est égal àfr. Remarquez que fr est le premier élément du tableau, même si de et us possèdent des indices inférieurs, car c'est le premier àavoir été affecté au tableau.

Les deux fonctions each() et list() peuvent être utilisées conjointement afin de parcourir un tableau, même si les indices ne sont pas séquentiels.

Exemple :

```
reset ($pays);
while (list ($cle, $valeur) = each ($pays)) {
   echo ("L'élément $cle est égal à$valeur <br>");
}
```

Cela signifie : Pour chaque élément du tableau, affectez comme valeur à $cle celle de l'indice (ou de la clé) de l'élément, et à$valeur la valeur de celui-ci.

Enfin, il existe une série de fonctions permettant de se déplacer dans un tableau :

Fonction	Description
reset()	Remet le pointeur interne de tableau au début.
pos()	Retourne la valeur de l'élément courant d'un tableau
key()	Retourne l'indice de l'élément courant d'un tableau.
next()	Avance le pointeur interne d'un tableau.
prev()	Recule le pointeur courant de tableau.
end()	Positionne le pointeur de tableau en fin de tableau.
sizeof()	Retourne le nombre d'éléments dans un tableau

Trier un tableau :

PhP dispose de plusieurs fonctions permettant de trier un tableau.

La plupart de ces fonctions ne retournent pas de valeur mais modifient simplement le tableau passé en paramètre.

Les fonctions ***sort()*** et ***rsort()*** permettent de trier un tableau par valeur croissante ou décroissante. Les indices sonts changés.

Exemple :

```
<?
$pays = array ("us" => "Etats-Unis",
               "ch" => "Suisse",
```

```
                "ca" => "Canada",
                   "fr" => "France",
                "de" => "Allemagne");

sort (($pays));

while (list ($cle, $val) = each ($pays)) {
   echo "L'élément $cle est égal à$val<BR>\n";
}
?>
```

Cet exemple imprime :
L'élément 0 est égal àCanada
L'élément 1 est égal àSuisse
L'élément 2 est égal àAllemagne
L'élément 3 est égal àFrance
L'élément 4 est égal àEtats-Unis

Cela réorganise le tableau afin que les valeurs soient en ordre alphabétique et que les indices reflètent également cet ordre.

On peut également trier un tableau par indice, àl'aide des fonctions ***ksort()*** et ***krsort(),*** qui s'utilisent de la même manière que les fonctions précédentes.

Exemple :

```
<?
$pays = array ("e" => "Etats-Unis" ,
                "d" => "Suisse",
                "c" => "Canada",
                "b" => "France",
                "a" => "Allemagne");
ksort (($pays));
while (list ($cle, $val) = each ($pays)) {
echo "L'élément  $cle est égal à $val<BR>\n";
}
?>
```

Cet exemple imprime :
L'élément a est égal àAllemagne
L'élément b est égal àFrance
L'élément c est égal àCanada
L'élément d est égal àSuisse
L'élément e est égal àEtats-Unis

La fonction usort() est un peu plus complexe. Elle prend un tableau comme argument, comme toutes les autres fonctions de tri, mais accepte également un second argument. Il s'agit d'une fonction définie par l'utilisateur et indiquant àusort() comment effectuer le tri. L'exemple ci-dessous trie le tableau selon la longueur de la chaîne contenue dans ses éléments.

Exemple :

```
<?
function par_taille ($a, $b) {
   $l_a = strlen ($a);
   $l_b = strlen ($b);
   if ($l_a == $l_b) return 0;
   return ($l_a < $l_b) ? -1 : 1;
}
```

```
$pays = array ("e" => "Etats-Unis" ,
                "d" => "Suisse",
                "c" => "Canada",
                "b" => "France",
                "a" => "Allemagne");

usort ($pays, par_taille);

while (list ($cle, $val) = each ($pays)) {
   echo "L'élément $cle est égal à$val<BR>\n";
}
?>
```

Ce code imprime :
L'élément 0 est égal àSuisse
L'élément 1 est égal àCanada
L'élément 2 est égal àFrance
L'élément 3 est égal àAllemagne
L'élément 4 est égal àEtats-Unis

La fonction de comparaison doit retourner un entier, qui sera inférieur, égal ou supérieur à zéro suivant que le premier argument est considéré comme plus petit, égal ou plus grand que le second argument. Si les deux arguments sont égaux, leur ordre est indéfini.

Chaînes de caractères et expressions régulières :

Fonctions de chaînes de caractères :

Nous allons détailler ci-dessous les fonctions de traitement de chaînes de caractéres les plus importantes de PhP.

- ***substr() :***

string ***substr***(string source, int début, int [taille])
Retourne une portion de ***strin***g, spécifiée avec le début ***début*** et la longeur ***taille***.

Exemple :

Echo(substr(Christophe),-2);
Imprime : he
Echo(substr(Christophe),-5,3);
Imprime : top
=> Renvoie une chaîne de trois caractères débutant au cinquième caractère en partant de la fin.
Echo(substr(Christophe),-6,-3);
Imprime : sto
=>Du sixième avant la fin au troisième avant la fin.
Lorsqu'une taille négative est spécifiée, la chaîne renvoyée se terminera à cette distance de la fin de la chaîne source.

- ***trim() :***

string ***trim*** (string ***str***)
Cette fonction retire les espaces blancs de début et de fin de chaîne, et retourne la chaîne nettoyée. Les espaces blancs sont : "\n", "\r", "\t", "\v", "\0", et " " (espace).

Exemple :

echo(***trim(*** chaîne exemple ***)***) ;
//imprime 'chaîne vide'

- ***strlen() :***

int ***strlen*** (string str)
Retourne la longueur de la chaîne string.

Exemple :
if(***strlen(***$userinput***)*** >30) {
echo(La saisie ne doit pas dépasser 30 caractères);
}

- ***implode()***

string ***implode*** (string *separator*, array *tableau*)
Retourne une chaîne constituée de tous les éléments du tableau, pris dans l'ordre, transformés en chaîne, et séparés par *séparateur*.
Exemple :

- ***explode()***

array ***explode*** (string separator, string string)
Retourne un tableau qui contient les éléments de la chaîne string, séparés par *separato*r.
Exemple :

```
<?php
$chaine="Nom|Prenom|Adresse";
$champs=explode("|",$chaine);
$boucle=0;
while($boucle<sizeof($champs)){
echo$champs[$boucle];
$boucle++;
}
?>
```

- ***str_replace()***

string ***str_replace***(modèle, remplacement, chaîne)
Remplace toutes les occurrences de modèle dans chaîne par remplacement

Exemple :

```
<?php
$chaine="tout est rouge";
$chaine=str_replace("rouge", "bleu", $chaine);
echo $chaine;
?>
```

Imprime : tout est bleu

Les expressions régulières :

(Source : PhPFrance, http://www.phpfrance.com)

Une expression régulière permet de caractériser le format d'une chaîne de caractère. Plus précisément, une expression régulière est un motif ou une série de motifs de caractères.

Le but étant de rechercher ou de remplacer un motif dans une chaîne de caractère.

Syntaxe d'une expression régulière :

Une expression régulière est une chaîne de caractères contenant des caractères spéciaux et des caractères standards. Il existe une grande variété de caractères spéciaux, le plus facile est de les voir au fur et àmesure àl'aide d'exemples.

- Les symboles **^** et **$** indiquent le début ou la fin d'une chaîne, afin de la délimiter

Exemples :

"**^debut**": chaine qui commence par *"debut"*
"**fin$**": chaine qui se termine par *"fin"*
"**^chaine$**": chaine qui commence et se termine par *"chaine"*
"**abc**": chaine contenant la chaine *"abc"*

- Les symboles *****, **+** et **?**, respectivement *"zero ou plusieurs"*, *"un ou plusieurs"*, *"un ou aucun"*, permettent de donner une notion de nombre.

Exemples :

"abc**+**": chaine qui contient "*ab"* suivie de un ou plusieurs *"c"* (*"abc"*, *"abcc"* etc..)
"abc*****": chaine qui contient "*ab"* suivie de zero ou plusieurs *"c"* (*"ab"*, *"abc"* etc..)
"abc**?**": chaine qui contient "*ab"* suivie de zero ou un *"c"* (*"ab"* ou *"abc"*)
"**^**abc**+**": chaine qui commence par "*ab"* suivie de un ou plusieurs *"c"* (*"abc"*, *"abcc"* etc..)

- Les accolades **{X,Y}** permettent de donner des limites de nombre.

Exemples :

"abc**{2}**": chaine qui contient "*ab"* suivie de deux *"c"* (*"abcc"*)
"abc**{2,}**": chaine qui contient "*ab"* suivie de deux *"c"* ou plus (*"abcc"* etc..)
"abc**{2,4}**": chaine qui contient "*ab"* suivie 2, 3 ou 4 *"c"* (*"abcc"* .. *"abcccc"*)

A noter que le premier nombre de la limite ("{0,2}", et pas "{,2}") est obligatoire. Les symboles vus précédemment ('*', '+', and '?') sont équivalents à"{0,}", "{1,}", et "{0,1}".

- Les parenthèses **( )** permettent de représenter une séquence de caractères.

Exemple :

"a**(**bc**)***": chaîne qui contient "*a"* suivie de zéro *"bc"* ou plus

- La barre verticale **|** se comporte en tant qu'opérateur OU.

Exemple :

"un**|**le": chaîne qui contient "*un"* ou *"le"*
"**(**un**|**le**) chien**": chaîne qui contient "*un chien"* ou *"le chien"*
"**(**a**|**b**)***": chaîne qui contient une suite de *"a"* ou de *"b"*

- Le point **.** indique n'importe quel caractère (une fois)

Exemple :

"**^.{**3**}$**": chaîne qui contient 3 caractères

- Les crochets **[ ]** définissent une liste de caractères autorisés (ou interdits). Le signe **-** permet, quant àlui, de définir un intervalle. Le caractère **^** après le premier crochet indique une interdiction.

Exemples :

"**[**abc**]**": chaîne qui contient un "*a"*, un *"b"*, ou un *"c"*
"**[**a**-**z**]**": chaîne qui contient un caractère compris entre "a" et "z"
"**[^**a**-**z**-**Z**]**": chaîne qui ne commence **pas** par une lettre

- Pour rechercher un caractère faisant partie des caractères spéciaux, il suffit de le faire précéder d'un antislash (un antislash doit donc être doublé), **SAUF** entre crochets.

En effet, dans les crochets, chaque caractère représente ce qu'il est. Pour représenter un **]** il faut le mettre en premier (ou après un **^** si c'est une interdiction), un **-** se met en premier ou en dernier.

Exemples :

"**[\+?{}.]**": chaîne qui contient un de ces six caractères
"**[]-]**": chaîne qui contient le caractère "]" ou le carcatère *"-"*

Les classes de caractères :

Dans les applications Internet, les expressions régulières sont particulièrement pratiques pour valider les saisies utilisateur (Numéro de téléphone, adresse électronique,…). Bien évidemment, cela est impossible àl'aide d'une recherche littérale : Il faut disposer d'un moyen permettant de décrire plus librement les valeurs àidentifier, et c'est justement le but des classes de caractère dont la syntaxe est : **[:classe:]**.

Classes de caractères	
[:alnum:]	caractère alphanumérique (équivalent à[A-Za-z0-9])
[:alpha:]	caractère alphabétique ([A-Za-z])
[:blank:]	caractère blanc (espace, tabulation)
[:ctrl:]	caractère de contrôle (les premiers du code ASCII)
[:digit:]	chiffre ([0-9])
[:graph:]	caractère d'imprimerie
[:print:]	caractère imprimable (tout sauf les caractères de contrôle)
[:punct:]	caractère de ponctuation
[:space:]	caractère d'espacement
[:upper:]	caractère majuscule
[:xdigit:]	caractère hexadécimal

Exemples :

"**^[:alnum:]+$**": chaîne composée d'un (ou plusieurs) caractère(s) alphanumérique(s)
"**[:punct:]|[:space:]**": chaîne contenant un caractère de ponctuation ou un caractère d'espacement
"**^[:digit:]+$**": nombre

Fonctions utilisant les expressions régulières :

Il existe plusieurs fonctions PHP, permettant de manipuler des chaînes de caractères àl'aide d' expressions régulières.

- Les fonctions **ereg()** et **eregi() :**

booleen ***ereg(***chaine expression, chaine texte[, tableau occurences]***)***

Permet d'évaluer le texte passé en paramètre, àpartir de l'expression régulière *expression.* Les occurences trouvées sont optionnellement stockées dans le tableau passé en paramètre. La fonction retourne la valeur **1** si une occurence est trouvée, et la valeur **0** si rien n'est trouvé.

La fonction "eregi" est identique àla fonction "ereg", mais ne tient pas compte de la casse (de la différence entre majuscules et minuscules).

Exemple :

```
if(ereg(   ^.+@.+\..+$   , $email)) {
        echo (   Adresse email valide   ) ;
}else{
        echo (   Adresse email erronée   ) ;
}
```

booleen ***eregi(***chaine expression, chaine texte[, tableau occurences]***)***

Cette fonction est identique à***ereg(),*** hormis le fait qu'elle ignore la casse des caractère lors de la recherche sur les caractère alphabétiques.

- Les fonctions **ereg_replace()** et **eregi_replace() :**

chaine ***ereg_replace(***chaine expr, chaine remplacement, chaine texte***)***

Retourne la chaîne texte passée en paramètre, modifiée àl'aide de l'expression régulière et de la chaîne de remplacement.

Exemple :

```
$str =    C'est alors que le fakir    ;
$pat =    fakir   ;
$repl =    FAKIR    ;
echo(ereg_replace($pat, $repl, $str)) ;
```

Ce code imprime :

C'est alors que le FAKIR

Comment remplacer les caractères non alphanumérique d'une chaine par un caractère particulier ?

```
<?php
$chaine = "Ceci # est # un test";
$chaine = eregi_replace("[^A-Z0-9]", "_", $chaine);
print $chaine;
?>
```

Et si vous désirez ne pas remplacer les espaces par des underscores,

```
<?php
$chaine = "Ceci # est # un test";
```

```
$chaine = eregi_replace("[^A-Z0-9\ ]", "_", $chaine);
print $chaine;
?>
```

Comment nettoyer les tags texte en préservant le texte ?

```
$chaine = eregi_replace("<font[^>.]*>","",$chaine);
$chaine = eregi_replace("</font[^>.]*>","",$chaine);
```

Ou $chaine désigne la chaine ànettoyer.

Les structures de contrôle

Tous les scripts PHP sont une suite d'instructions. Une instruction peut être une assignation, un appel de fonction, une instruction conditionnelle ou bien une instruction qui ne fait rien (une instruction vide).

Une instruction se termine habituellement par un point virgule (";"). De plus, plusieurs instructions peuvent être regroupées en bloc, délimité par des accolades ("{}"). Un bloc est considéré comme une instruction.
Les instructions conditionnelles (structures conditionnelles) procurent aux programmes les capacités décisionnelles nécessaires àl'accomplissement de la plupart des tâches informatqiues. Nous détaillons ci-dessous la syntaxe des principales structures de contrôle.

Les structures conditionnelles :

Les structures conditionnelles permettent de définir des blocs de code qui ne seront exécutés que si certaines conditions sont remplies.

Instruction If :
Elle permet l'exécution conditionnelle d'une partie de code. Les fonctionnalités de l'instruction if sont les mêmes en PHP qu'en C : Si la condition renvoie false, PhP permet d'exécuter un autre bloc de code àl'aide du mot clé else.

if (condition){
expression vrai
} else {
expression faux
}

PhP dispose également du mot clé elseif : Comme l'expression else, elle permet d'exécuter une instruction après un if dans le cas oú le "premier" if est évalué comme FALSE. Mais, àla différence de l'expression else, il n'exécutera l'instruction que si l'expression condionnelle elseif est évaluée àTRUE.
Une instruction if peut comporter un nombre quelconque d'instructions elseif. La branche finale else permet alors de spécifier le code devant être exécuté si aucune des conditions précédentes n'est remplie.

Exemple :

```
<? php
$a = 10; $b = 11;
print "a est égale à$a, b est égale à$b. ";
if ($ a > $b) {
print "a est plus grand que b";
} elseif ($ a == $b) {
print "a est égal àb";
} else {
print "==> a est inférieure àb.";
}
?>
```

PhP propose également une autre syntaxe pour l'instruction if : ***if :..endif ;***

L'exemple suivant illustre cette alternative.
Exemple :

```
<? php
$a = 10; $b = 11;
print "a est égale à$a, b est égale à$b. ";
if ($ a > $b) :
print "a est plus grand que b";
elseif ($ a == $b) :
print "a est égal àb";
else :
print "==> a est inférieure àb.";
endif ;
?>
```

Dans l'exemple, les accolades ne sont pas utilisées et les conditions testées sont suivies d'un deux points (:).
La dernière endif sert àsignaler la fin du bloc if, remplaçant une accolade fermante}.

Les structures itératives :
Les structures itératives (boucles) constituent un moyen d'exécuter un bloc de code un nombre de fois donné, où jusqu'àce qu'une condition particulière soit satisfaite.
PhP possède deux types de boucles :

- Une boucle ***While*** : Teste une condition avant ou après chaque itération et ne parcourt de nouveau la boucle que si la condition est vérifiée.
- Une boucle ***for*** : Le nombre d'itérations est fixé une fois pour toutes, avant l'exécution de la première boucle.

Boucle While :

```
while (condition){
        //instructions
}
```

La signification d'une boucle while est très simple. Le PHP exécute l'instruction tant que l'expression de la boucle while est évaluée comme TRUE. La valeur de l'expression est vérifiée àchaque début de boucle, et, si la valeur change durant l'exécution de l'instruction, l'exécution ne s'arrêtera qu'àla fin de l'itération (chaquefois que le PHP exécute l'instruction, on appelle cela une itération).
Si l'expression du while est FALSE avant la première itération, l'instruction ne sera jamais exécutée.
Pour stopper l'éxécution de façon précoce, on peut utiliser l'instruction ***break***.

Exemple :

```
<?php
/* Afficher tous les nombres de 1 à10 */
$i = 1;
while ($i <= 10) {
   print $i++;  /* La valeur affichée est $i avant l'incrementation
                (post-incrementation)  */
}
?>
```

Comme l'instruction if, while accepte une autre syntaxe en utilisant les deux points (:)

Exemple :

```
<?php
/* Afficher tous les nombres de 1 à10 : Autrement*/
```

```
$i = 1;
while ($i <= 10):
   print $i;
   $i++;
endwhile;
?>
```

Boucle do.. While :
Les instructions do..while sont similaires aux instructions while, si ce n'est que la condition est vérifiée àla fin de chaque itération, et non au début. Cela signifie que la boucle s'exécute toujours au moins une fois (l'expression n'est testée qu'àla fin de l'itération), ce qui n'est pas le cas lorsque vous utilisez une boucle while (l'expression est vérifiée au début de chaque itération).
Exemple :

```
$i = 0;
do {
print $i;
} while ($i>0);
```

La boucle ci-dessus ne va être exécutée qu'une seule fois, car lorsque l'expression est évaluée, elle vaut FALSE (car la variable $i n'est pas plus grande que 0) et l'exécution de la boucle s'arrête.

Boucle For :

```
for (expr1; expr2; expr3) {
instructions
}
```

L'instruction accepte trois expressions :

- La première expression *(expr*1) est évaluée (exécutée) quoi qu'il arrive au début de la boucle.
- Au début de chaque itération, l'expression *expr2* est évaluée. Si l'évaluation vaut TRUE, la boucle continue et l'instruction est exécutée. Si l'évaluation vaut FALSE, l'exécution de la boucle s'arrête.
- A la fin de chaque itération, l'expression *expr3* est exécutée.

L'expression expr2 compare en général la variable de contrôle de boucle àune valeur prédéfinie, mais cela n'a rien d'obligatoire.

Exemple :

```
for ($i=0 ; $i<10 ; $i++)
 {
   echo "$i";
 }
```

PHP supporte aussi la syntaxe alternative suivante pour les boucles for :

```
for (expr1; expr2; expr3):
instructions; ...;
endfor;
```

Exemple :

```
for($i=0 ; $i<10 ; $i++) :
echo "$i";
endfor ;
```

Les structures de branchement :

switch (expression***) {***
case resultat1 ***:***
instructions àexécuter si l'expression vaut resultat1...
break ;
case resultat2 ***:***
instructions àexécuter si l'expression vaut resultat2...
break ;
...
default :
instructions àexécuter en dernier recours...
}

L'instruction switch équivaut àune série d'instructions if..elseif..elseif....else ;
Elle est utilisée pour comparer la même variable (ou expression) avec un grand nombre de valeurs différentes et d'exécuter différentes parties de code suivant la valeur àlaquelle elle est égale.
L'instruction ***break*** a pour but de stopper l'exécution des instructions d'une clause case donnée

Exemple :
Switch ($type***) {***
case "Femme" ***:***
echo "Bonjour Madame" ;
break ;
case "Homme" ***:***
echo "Bonjour Monsieur" ;
break ;
default :
echo "Bonjour, vous êtes bizarre !" ***;***
}

Les fonctions

Une fonction est un bloc de code défini une fois, puis invoqué à partir d'autres endroits du programme.
Une fonction peut accepter un ou plusieurs arguments, accomplit un ensemble d'opérations prédéfinis selon ces arguments puis renvoie une valeur en résultat.

Une fonction peut être définie en utilisant la syntaxe suivante :

```
function ma_fonction ($arg_1, $arg_2, ..., $arg_n)
{
# Corps de ma fonction...
return $valeur_de_retour ;
}
```

En PHP toute fonction doit être préalablement définie avant d'être utilisée.
Les valeurs sont renvoyées en utilisant une instruction de retour optionnelle. Tous les types de variables peuvent être renvoyés (tableaux et objets compris).

Exemple :

```
function carre($nombre) {
return $nombre * $nombre ;
}
echo carre(3) ; # affiche la valeur 9
```

Les arguments de fonction :

Les arguments constituent un moyen de transmettre des données à la fonction.
Par défaut les arguments sont transmis ***par valeur*** : Cela signifie que la variable paramètre interne à la fonction contient une copie de la valeur qui lui a été transmise. Si la valeur du paramètre change, cela ne modifie pas la valeur de la variable.

Exemple :

```
function imprime_double($n) {
        $n = $n * 2 ;
        echo($n) ;
}

$a = 5 ;
echo ( $a ) ; //imprime 5
imprime_double($a) ; //imprime 10
echo($a) ; //imprime de nouveau 5
```

La valeur de $a n'est pas modifiée, même après avoir été transmise à imprime_double(), puisque l'argument a été transmis comme valeur.
En revanche, si un argument est transmis *par référence*, toute modification apportée à la variable paramètre *modifie* la variable de l'instruction d'appel.
Pour passer un argument par référence, il faut ajouter un '&' devant l'argument dans la déclaration de la fonction.

Exemple :

```
function augmentation(&$salaire, $pourcent) {
        //augmente $salaire de $pourcent
        $salaire+=$salaire * $pourcent/100 ;
}
```

```
$sal = 50000 ;
echo(   Salaire avant augmentation : $sal) ;            //Imprime 50000
augmntation($sal, 4) ;
echo (   Salaire après augmentation : $sal<BR>\n   ) ;       //Imprime 52000
```

Si vous souhaitez passer une variable par référence àune fonction mais de manière ponctuelle, vous pouvez ajouter un '&' devant l'argument dans l'appel de la fonction.

Exemple :

```
function augmentation($salaire, $pourcent) {
        //augmente $salaire de $pourcent
        $salaire+=$salaire * $pourcent/100 ;
}

$sal = 50000 ;
echo(   Salaire avant augmentation : $sal) ;            //Imprime 50000
augmntation(&$sal, 4) ;
echo (   Salaire après augmentation : $sal<BR>\n   ) ;       //Imprime 52000
```

Portée d'une variable :

La portée d'une variable détermine quelles parties du programme y ont accès.

Pour la plupart des variables, la portée concerne la totalité d'un script PHP. Mais, lorsque vous définissez une fonction, la portée d'une variable définie dans cette fonction est locale à la fonction.

Exemple :

```
$var=   a   ;
function change_var() {
  $var =    b    ;
}
change_var() ;
echo(   $var   ) ;  //affiche    a
```

Ce code imprime a . La variable $var àl'intérieur de la fonction possède une portée locale, et est de ce fait différente de la variable $var externe àla fonction.

La variable extérieure àla fonction possède une portée globale, car elle peut être atteinte et modifiée par tout fragment du code du script général, hors d'une fonction.

Deux façons existent pour accéder àune variable globale au sein d'une fonction :
– Déclarer la variable comme *global* au sein de la fonction,
- Utiliser le tableau associatif *$GLOBALS* avec comme clé le nom de la variable globale.

Exemple :

```
$var=   a   ;
function change_var() {
        global $var ;
        $var =    b    ;
}
change_var() ;
echo(   $var   ) ;  //affiche    b
```

Le code imprime maintenant b , parce que $var fait référence àla même variable, à l'intérieur et àl'extérieur de la fonction.

On peut également avoir rcours au tableau intégré *$GLOBALS*, qui contient toutes les variables globales du script :

Exemple :

```
$var= a ;
function change_var() {
        $GLOBALS[ var ] = b ;

}
change_var() ;
echo( $var ) ; //affiche b
```

Remarquez que l'opérateur $ ne figure pas avant le mot var lors de l'utilisation de ce tableau.

Variables statiques :

Une variable statique est une variable locale qui ne perd pas sa valeur àchaque fois que la fonction est exécutée. On utilise, comme en C, l'attribut ***static*** pour déclarer une telle variable.
Ce type de variable est très utile pour la création de fonctions récursives.

Exemple :

```
function compte () {
static $compteur = 0 ;
$compteur++ ;
echo "$compteur " ;
if ($compteur < 10) compte() ;
}
compte() ;
```

Lors du tout premeir appel de la fonction compte(), la variable statique $compteur va être créée et se voir affecter la valeur spécifiée (ici 0), mais par la suite elle conservera sa valeur entre chaque appel de la fonction : La variable ne sera pas réinitialisée.

Lorsque vous souhaitez qu'une fonction retourne plusieurs valeurs, le plus simple est d'utiliser un tableau :

```
<?
function nom_fonction()
{
.....
return array( $variable1, $variable2, $variable3 ); // on retourne les valeurs voulues dans un tableau
}
$retour = nom_fonction();
echo "$retour[0] - $retour[1] - $retour[2]";
?>
```

Traitement des formulaires en PhP

Le but de ce chapitre est de présenter une méthodologie de traitement des formulaires dans le cas d'un site web dynamique.

La structure générale d'un script de traitement de formulaire doit :

- Implémenter le formulaire lui même ;
- Procéder àdes contrôles de saisie ;
- Effectuer des actions suivant que la saisie est valisée ou non

Squelette d'un script de traitement de formulaire :

```
 <html>
<body>
…
< ?php
//Est-ce la première fois qu'on passe dans le script
//NON
if(NON) {
//contrôle sur la saisie
….
//Si la saisie est correcte : SAISIE VALIDE
if(SAISIE VALIDE) {
// Actions diverses
…
// Message
…
}
//SINON : Saisie INVALIDE
else{
//Actions diverses
…
//Message
..
}
}
//OUI ou saisie invalide
if(OUI ou SAISIE INVALIDE) {
?>
<form method=  POST  action=  script.php  >
…
<INPUT TYPE=  HIDDEN  NAME=  formulaire  value=  ok  >
</FORM>
<?
}
?>
…
</BODY>
</HTML>
```

Au niveau de la ligne : <*INPUT TYPE= HIDDEN NAME= formulaire value= ok* >, on a utilisé un champ caché (HIDDEN) permettant de déterminer si on a déjà passé ou non dans le formulaire.
Le principe consiste à tester, au début du script, si la valeur associée au champ caché formulaire est bien initialisée ou non : si oui, c'est que nous sommes déjà passé dans le formulaire. Sinon, c'est que le formulaire est chargé pour la première fois et qu'il n'y a donc aucun traitement particulier à faire.

Exemple :

Nous allons étudier un exemple concret en commençant par définir les objectifs de notre formulaire.

Admettons que nous devions élaborer un formulaire assurant **la saisie du nom, prénom et de l'adresse e-mail** d'un visiteur, afin de les **stocker dans une base de données**.
Deux champs seront en **saisie obligatoire : nom et adresse e-mail**. Et dans une certaine mesure, nous chercherons à **contrôler la validité de l'adresse e-mail** saisie.

Pour finir, notre formulaire devra **renvoyer un message au client afin de l'inviter à modifier sa saisie si cette dernière est invalide, ou de l'informer que sa saisie a été prise en compte dans le cas contraire**.

Le code source correspondant est le suivant :

```
<html>
<body>
<?php
if($formulaire)
{
    $echec="";

    if($nom=="")
    $echec=$echec."- Le champ nom est obligatoire<br>";

    if($email=="")
    $echec=$echec."- Le champ email est obligatoire";

    elseif (!ereg("^(.+)@(.+)\\.(.+)$",$email))
    $echec=$echec."- Le champ email est invalide";

    if($echec=="") {
        //traitement a effectué dans le cas de succéè
        $message="Succès.<br>Votre saisie a été prise en
comptre.<br>Merci.";
        print $message;
    }
    else {
        $message="Echec.<br>Corrigez votre saisie.<br>".$echec;
        print $message;
    }
}
if(!$formulaire || $echec != "") {
?>
<form method="post" action="script.php">
Nom
<input type="text" name="nom" value="<? Print $nom ?>">
<br>
```

```
Prénom
<input type="text" name="prenom" value="<? Print $prenom ?>">
<br>
Adresse email
<input type="text" name="email" value="<? Print $email ?>">
<br>
<input type="submit" value="Valider">
<input type="reset" value="Effacer">
<input type="hidden" name="formulaire" value="ok">
</form>
<?php
}
?>
</body>
</html>
```

Comment traiter les champs texte :

```
<html>
<body>
<?php
if($chaine)
   echo "chaine : $chaine";
?>
<form method=post action="texte.php">
<input type=text name=chaine size=16 maxlength=255 value="<?php echo
htmlentities($chaine) ?>">
<br>
<input type=submit value=Valider>
</form>
</body>
</html>
```

Comment traiter les champs textearea :

```
<html>
<body>
<?php
if($texte)
echo "text : <br>".nl2br($texte);
?>
<form method=post action="textarea.php">
<textarea name=texte cols=20 rows=10><?php echo $texte ?></textarea>
<br>
<input type=submit value=Valider>
</form>
</body>
</html>
```

Comment traiter les cases à cocher

```
<html>
<body>
<?php
if($choix1)
   echo "Choix 1 : Oui";
else
   echo "Choix 1 : Non";
echo " / ";
if($choix2)
   echo "Choix 2 : Oui";
else
   echo "Choix 2 : Non";
?>
<form method=post action="case.php">
<input type="checkbox" name="choix1" value="Oui" <?php if($choix1) echo
"checked" ?>>Choix 1
<br>
<input type="checkbox" name="choix2" value="Oui" <?php if($choix2) echo
"checked" ?>>Choix 2
<br>
<input type=submit name=submit value=Valider>
</form>
</body>
</html>
```

Comment traiter les boutons radio

```
<html>
<body>
<?php
if($choix)
   echo "Choix : $choix";
?>
<form method=post action="bouton.php">
<input type="radio" name="choix" value="Oui" <?php if($choix=="Oui") echo "checked" ?>>Oui
<br>
<input type="radio" name="choix" value="Non" <?php if($choix=="Non") echo "checked" ?>>Non
<br>
<input type="radio" name="choix" value="Peut-être" <?php if($choix=="Peut-être") echo "checked"
?>>Peut-être
<br>
<input type=submit name=submit value=Valider>
</form>
</body>
</html>
```

Comment traiter les listes

L'exemple suivant vous y aidera peut-être :

```
<html>
<body>
<?php
if($liste)
   echo "Choix : $liste";
?>
```

```
<form method=post action="liste.php">
<select name=liste size=1>
<option value="choix1" <?php if($liste=="choix1") echo "selected" ?>>Choix 1
<option value="choix2" <?php if($liste=="choix2") echo "selected" ?>>Choix 2
<option value="choix3" <?php if($liste=="choix3") echo "selected" ?>>Choix 3
<br>
<input type=submit name=submit value=Valider>
</form>
</body>
</html>
```

Comment traiter les listes multiples

```
<html>
<body>
<?php
if($liste) {
   while(list($key,$val)=each($liste)) {
      echo "Choix : $val";
      echo "<br>";
   }
}
else
   $liste=array();
?>
<form method=post action="multiple.php">
<select name=liste[] size=3 multiple>
<option value="choix1" <?php if(in_array("choix1",$liste)) echo "selected" ?>>Choix 1
<option value="choix2" <?php if(in_array("choix2",$liste)) echo "selected" ?>>Choix 2
<option value="choix3" <?php if(in_array("choix3",$liste)) echo "selected" ?>>Choix 3
<option value="choix4" <?php if(in_array("choix4",$liste)) echo "selected" ?>>Choix 4
<option value="choix5" <?php if(in_array("choix5",$liste)) echo "selected" ?>>Choix 5
<br>
<input type=submit name=submit value=Valider>
</form>
</body>
</html>
```

Attention :
Ce code n'est qu'un exemple à titre indicatif et pédagogique. Le but n'est en aucun cas de proposer une solution optimisée, souvent fortement dépendante de la problématique réelle à traiter.

Utiliser MySQL avec PhP

Nous détaillons dans ce chapitre l'usage du PhP avec les bases de données et plus précisément la base MySQL.

Le langage SQL ne sera pas traité en détail dans ce chapitre. Nous abordons essentiellement les fonctions PhP permettant d'accéder et de manipuler les bases de données MySQL.

L'utilisation de MySQL avec PhP s'effectue en quatre étapes :

- *Connexion au serveur des données ;*
- *Sélection de la base des données ;*
- *Requête ;*
- *Exploitation des requêtes.*

Connexion au serveur des données :

Pour se connecter au serveur des bases de données , on peut faire appel àl'une des fonctions suivantes :

- int ***mysql*_connect *(***string hostname :port :connexion , string username , string password***)***

  Permet d'ouvrir une connexion simple.
  Pour se connecter, il faut définir l'adresse du serveur de données ainsi que le nom d'utilisateur et le mot de passe. La valeur par défaut de hostname est localhost .
  La fonction mysql_connect() retourne un identifiant. Si cet identifiant est égal à0, alors une erreur s 'est produite pendant la phase de connexion. Si l'identifiant est différent de 0, tout c'est bien passé et on peut se connecter au serveur des bases de données.

  Exemple :

```
<?
 if( mysql_connect( 'localhost' , 'moi' , 'miaou' ) > 0 )
  echo 'Connexion réussie !' ;
 else
  echo 'Connexion impossible !' ;
?>
```

- int ***mysql_p*connect *(***string hostname :port :connexion , string username , string password***)***
  Les arguments et et la valeur de retour sont les mêmes que musql_connect().
  Les différences entre mysql_connect() et mysql_pconnect() :
  Premièrement, lors de la connexion, la fonction essaie de trouver une connexion permanante déjàouverte sur cet hote, avec le même nom d'utilisateur et de mot de passe. Si une telle connexion est trouvée, son identifiant est retourné, sans ouvrir de nouvelle connexion.
  Deuxièmement, la connexion au serveur MySQL ne sera pas terminée avec la fin du script. Au lieu de cela, le lien sera conservé pour un prochain accès (mysql_pconnect() établit une connexion persistante àun serveur MySQL.).

Sélection de la base :

Les étapes sélection et requête peuvent être faites en même temps, mais il est plus simple si vous utilisez une seule base, de la sélectionner avant de commencer vos requêtes. Ainsi, toutes les requêtes àvenir utiliseront cette base par défaut.

- int **mysql_select_db** (string *database_nam*e, int *link_identifier*)
  Le paramètre database_name est obligatoire, le paramètre link_identifier facultatif. La fonction retourne true ou false selon que l'opération réussit ou non.
  Si on ne donne pas le paramètre *link_identifier*, la fonction utilise la dernière connexion ouverte.
  Toutes les instructions passées au serveur MySQL seront exécutées sur la base de données active.

  Exemple :

```
<?
 if( mysql_select_db( 'ma_base' ) == True )
  echo 'Sélection de la base réussie' ;
 else
  echo 'Sélection de la base impossible' ;
?>
```

Requête :

Pour interroger la base de données, on utilise l'une des fonctions suivantes :

- int **mysql_query** (string query, int link_identifier)
  Envoie au serveur MySQL une instruction SQL àexécuter.
  Le paramètre query est obligatoire. Si le paramètre link_identifier n'est pas précisé, la dernière connexion est utilisée. Si aucune connexion n'a été ouverte, la fonction tentera d'en ouvrir une, avec la fonction mysql_connect() mais sans aucun paramètre (c'est àdire avec les valeurs par défaut).

  Comme pour la connexion, cette fonction retourne un identifiant qui est à"0" lorsqu'une erreur s'est produite.
  Contrairement aux autres fonctions (connexion et sélection), cet identifiant est très important, il est utilisé pour retrouver les données rapatriées par une requête de sélection.

  Lorsque l'on fait une requête de sélection, MySQL retourne les données et PHP les place en mémoire. L'identifiant permet donc de retrouver ces données en mémoire. En fin de script, la mémoire est libérée et les données sont donc perdues. Il est possible de libérer la mémoire, pour ne pas surcharger le serveur, avant la fin du script au moyen de la commande mysql_free_result.

  Exemple :
  On part du principe que la base de données est déjàsélectionnée et on veut avoir des informations sur le membre nommé "alain" :

```
<?
 $requete = "SELECT * FROM membres WHERE nom = 'alain' ";
 $resultat = mysql_query( $requete );
?>
```

- int **mysql_db_query** (string database, string query, int link_identifier)

Envoie au serveur MySQL une instruction MySQL contenant le nom de la base de données active.
Les paramètres database et query sont obligatoires ; le paramètre link_identifier est facultatif.
La différence par rapport àmysql_query(), c'est qu'on peut sélectionner la base des données en même temps que l'exécution de la requête.

Exemple :

```
$req= "SELECT * from employee";
$result = mysql_db_query("database1", $req, $linkId)
```

Exploitation des requêtes :

Requête de sélection :

A la suite d'une requête de sélection, les données sont mises en mémoire.
Pour pouvoir les exploiter, PhP gère un pointeur de résultat, c'est àdire qu'il repère un enregistrement parmi les autres, et lorsque l'on veut en lire un, c'est celui qui est pointé qui sera retourné et le pointeur est déplacé vers l'enregistrement suivant.
Les fonctions de lecture du résultat sont les suivantes :

- array **mysql_fetch_row** (int result)

Extrait de l'identificateur de résultat (*result*) la prochaine ligne sous la forme d'un *tableau énuméré* ou FALSE s'il ne reste plus de ligne.

Exemple :

```
< ?
$enregistrement = mysql_fetch_row($result);
//Afficher le champ nom
echo $enregistrement[0] ;
//Afficher le champ prénom
echo $enregistrement[1] ;
//Afficher le champ date
echo $enregistrement[2] ;
?>
```

- array **mysql_fetch_array** (int result, int result_type)

Extrait la ligne sous forme d'un *tableau associatif*.
Le paramètre result_type est facultatif.Il peut prendre les valeurs suivantes :
MYSQL_NUM : Le tableau ne contient que des indices numériques (effet similaire à celui de mysql_fetch_row)
MYSQL_ASSOC : Le tableau ne contient que des indices associatifs.
MYSQL_BOTH : Le tableau contient àla fois des indices numériques et des indices associatifs.
Si l'argument result_type n'a pas été spécifié, MYSQL_BOTH est considérée comme valeur par défaut de cet argument.

Exemple :

```
< ?
$enregistrement = mysql_fetch_array($result);
//Affiche le champ –prenom-
echo $enregistrement['prenom'] ;
//Affiche le champ –nom-
echo $enregistrement['nom'] ;
//Affiche le champ –date-
echo $enregistrement['date'] ;
```

?>

- object **mysql_fetch_object** (int *resul*t, int *result_typ*)

Retourne un objet (structure) correspondant àla ligne extraite de l'identificateur de résultat.
L'argument optionnel ***result_typ*** est une constante qui peut prendre les valeurs suivantes : MYSQL_ASSOC, MYSQL_NUM, et MYSQL_BOTH.

Exemple :

```
< ?
$enregistrement = mysql_fetch_object($result);
//Affiche le champ –prenom-
echo $enregistrement->prenom ;
//Affiche le champ –nom-
echo $enregistrement->nom;
//Affiche le champ –date-
echo $enregistrement->date ;
?>
```

- int **mysql_num_rows** (int *resul*t)

Retourne le nombre d'enregistrements qui ont été retournés par la sélection.

Exemple :

On affiche le nombre d'enregistrement et on parcourt l'ensemble des données en les affichant au moyen d'une boucle :

```
< ?
$requete = "SELECT * FROM membres" ;
$result = mysql_query($requete);
echo "il y a" .mysql_num_rows($result) ." membre(s)";
while($enregistrement = mysql_fetch_array($result))
{
echo $enregistrement[‘nom’]. " " . $enregistrement[‘prenom’] ;
}

?>
```

Requête d'insertion :

Suite àune requête d'insertion, on peut récupérer la valeur d'un champ *auto incrémenté*. Ceci est faisable grâce àla fonction suivante :

- int **mysql_insert_id** (int *link_identifier*)

mysql_insert_id() retourne le dernier identifiant généré par un champs de type AUTO_INCREMENTED.
Cette fonction ne prend aucun argument. Elle retourne le dernier identifiant généré par la dernière fonction INSERT effectuée.

Exemple :

Suite àune requête d'insertion, on veut afficher le numéro auto incrémenté :

```
< ?
echo "Votre numéro d'identifiant est : " .mysql_insert_id() ;
?>
```

Requête de modification :

Lorsque l'on change les données en insérant, supprimant ou modifiant, il est possible de connaître le nombre d'enregistrements affectés par la requête. Ceci est faisable grâce àla fonction suivante :

- int **mysql_affected_rows** (int *link_identifier*)

Le paramètre *link_identifier* est facultatif. La fonction retourne le nombre de lignes traitées dans la requête SQL précédente.
Cette fonction permet de fixer le nombre de lignes insérées, mises àjour ou supprimées par la précédente requête SQL (INSERT, DELETE, REPLACE ou UPDATE) envoyée au serveur.

Exemple :
Suite àune commande "UPDATE", on voudrait savoir combien de lignes ont été modifiées :

```
< ?
echo mysql_affected_rows() ." eregistrement(s) modifiés";
?>
```

Etude de cas : (*Source : www.ilovephp.com*)

Ce cas vous montre comment utiliser MySQL et PHP, en se basant sur un exemple : un annuaire de personnes composé :
identifiant : chaîne de caractères de 10 caractères maximum, un identifiant est unique
nom : chaîne de caractères de 100 caractères maximum
prenom : chaîne de caractères de 100 caractères maximum
age : entier
dateajout : une date

Schéma de la table :

```
create table annuaire (
identifiant varchar(10) not null,
nom varchar(100),
prenom varchar(100),
age int(3),
dateajout date,
primary key(identifiant)
);
```

Tout d'abord, vous devez vous connecter àla base de données.

Il y a 4 paramètres : l'adresse du serveur où la base de données (BDD) est hébergée, le nom de l'utilisateur, le mot de passe et le nom de la base de données àutiliser.

```
$dbhost = "localhost"; //si l'ordinateur qui héberge la BDD est le même que le serveur HTTP
$user = "nom";
$password = "motdepasse";
$usebdd = "nomdelabdd";
```

La fonction php vous permettant de vous connecter est mysql_connect.

```
$connexion = mysql_connect("dbhost","user","password");
```

Pour vérifier si la connexion est valide :

```
if (!$connexion) {
echo "Impossible d'effectuer la connexion";
exit;
```

}

Ensuite, vous devez sélectionner la base sur laquelle vous voulez travailler !

$db = mysql_select_db("$usebdd", $connexion);

Pour vérifier si la base de données est bien sélectionnée

if (!$db) {
echo "Impossible de sélectionner cette base de données";
exit;
}

Nous allons ensuite voir comment insérer des données en rajoutant une ligne dans la table annuaire

sfim,airey,romuald,23
Voici la commande SQL le permettant
insert into annuaire values ('sfim','airey','romuald',23,curdate());
Il suffit ensuite d'utiliser la commande mysql_query, ce qui donne :
$resultat_sql = mysql_query("insert into annuaire values
('sfim','airey','romuald',23,curdate())",$connexion);

Pour complèter la table on répète l'opération :
insert into annuaire values ('sfim','airey','romuald',23,curdate());
insert into annuaire values ('orawat','rawat','olivier',24,curdate());
insert into annuaire values ('dmarc','dupuis','marc',32,curdate());
insert into annuaire values ('adupont','dupont','antoine',22,curdate());

Comment faire pour supprimer une ligne ?

la commande sql serait :
delete from annuaire where identifiant= 'dmarc';
De même on utilise la commande mysql_query
$resultat_sql = mysql_query("delete from annuaire where identifiant= 'dmarc'",$connexion);
ok maintenant on sait se connecter, insérer, effacer.

Comment récupère-t-on les données ?

la table contient :
sfim,airey,romuald,23,2000-01-15
orawat,rawat,olivier,24,2000-01-15
dmarc,dupuis,marc,32,2000-01-15
adupont,dupont,antoine,22,2000-01-15
Par exemple, je veux sélectionner le nom, le prénom et l'âge des personnes dans l'annuaire de moins de 30 ans et les ranger par âge décroissant :
voici la commande SQL :
select nom,prenom,age from annuaire where age < 30 order by age desc;
De même que pour l'insertion et la suppression j'utilise la commande mysql_query
$resultat_sql = mysql_query("select nom,prenom,age from annuaire where age <
30",$connexion);
Le résultat de cette commande est retourné dans $resultat_sql, il faut l'imaginer comme une table qui serait de cette forme

ligne	nom	prenom	age
0	dupont	antoine	22
1	airey	romuald	23
2	dupuis	marc	24

La première chose àsavoir est : Combien ai-je de lignes de réponse ?
$nombreligne = mysql_num_rows($resultat_sql);
Ici j'en aurai 3
Il ne reste plus qu'àrécupérer le résultat.

```
$i = 0;
while ($i<$nombreligne)
{
$enregistrement= mysql_fetch_row($resultat_sql);
echo "$enregistrement[0] $enregistrement[1] $enregistrement[2] ";
$i++;
}
```

Pour finir un récapitulatif !

```
<?php
// ATTENTION, pour que le script fonctionne la table doit être créée

$dbhost = "localhost";
$user = "nom";
$password = "motdepasse";
$usebdd = "nomdelabdd";

//connexion au serveur MySQL

$connexion = mysql_connect("dbhost","user","password");
if (!$connexion) {
echo "Impossible d'effectuer la connexion";
exit;
}

//sélection de la BDD

$db = mysql_select_db("$usebdd", $connexion);
if (!$db) {
echo "Impossible de sélectionner cette base données";
exit;
}

// insertion des données

$resultat_sql = mysql_query("insert into annuaire values
('sfim','airey','romuald',23,curdate())", $connexion);
$resultat_sql = mysql_query("insert into annuaire values
('orawat','rawat','olivier',24,curdate())", $connexion);
$resultat_sql = mysql_query("insert into annuaire values
('dmarc','dupuis','marc',32,curdate())", $connexion);
$resultat_sql = mysql_query("insert into annuaire values
('adupont','dupont','antoine',22,curdate())", $connexion);
```

```
// sélection des données

$resultat_sql = mysql_query("select nom,prenom,age from annuaire where age <
30",$connexion);

// nombre de lignes

$nombreligne = mysql_num_rows($resultat_sql);

// affichage du résultat dans une table HTML

echo "<table><tr><th>Nom</th><th>Prenom</th><th>Age</th></tr>";

$i = 0;
while ($i<$nombreligne)
{
$enregistrement= mysql_fetch_row($resultat_sql);
echo "$enregistrement[0] $enregistrement[1] $enregistrement[2] ";
$i++;echo "<tr><td>$nom</td><td>$prenom</td><td>$age</td></tr>";
$i++;
}

echo "</table>";
?>
```

Installation du serveur Web Apache avec PhP

Pour pouvoir tester les scripts PhP, on a besoin d'installer un serveur Web (Apache dans notre cas).
On peut installer Apache et PhP de deux façons :

- **Automatiquement**, en utilisant un utilitaire qui installe et configure automatiquement tout ce dont vous avez besoin (tels que EasyPHP : http://www.manucorp.com/easyphp.php3). C'est simple et rapide, mais vous ne savez pas ce qui se passe.
- **Manuellement**, en configurant les différents fichiers d'Apache et PhP (Le but de ce chapitre).

Installation d'Apache :

1. Récupérer la dernière version d'Apache (Site officiel : www.apache.org, sous la rubrique : Apache for Win32)

2. Le programme apche.exe est un wizard d'installation, c'est ce que vous voyez chaque fois que vous installez un programme ➔ *SETUP* : L'installation va vous demander un chemin d'installation - par défaut "C:\Program Files\Apache\Group\Apache" - je conseille de modifier celui ci en "C:\Apache" afin d'avoir un accès instantané aux fichiers du serveur.

   Une fois installé vous devez avoir une arborescence comme ceci :

Arborescence	**Désignation**
c:/Apache	Contient l'exécutable du serveur
c:/Apache/bin	Contient le programme permettant de créer des utilisateurs
c:/Apache/cgi-bin	Contient vos sources cgi-bin, exemple : le compteur, le guestbook,..
c:/Apache/conf	Contient le fichier de configuration (httpd.conf)
c:/Apache/htdocs	Contient le site web proprement dit (les fichiers html, les images, les fichiers destinés àêtre téléchargés ..)
c:/Apache/icons	Contient les icônes qui seront affichées lors des requêtes représentant le contenu d'un répertoire.
c:/Apache/log	Contient les fichiers d'accès au serveur – accès et erreurs.
c:/Apache/modules	Contient des modules additionnels au seveur comme Perl ou les Servlets (Java).
c:/Apache/src	Contient le programme source du serveur.

3. Une fois l'installation terminée, il faut éditer le fichier **httpd.conf** (qui se trouve dans le répertoire c:/Apache/ */conf)*

Au niveau de la section 2 *Main server configuration*, procéder aux modifications suivantes :

```
ServerName 127.0.0.1 //Pour faire un test en local
DocumentRoot "C:/Apache/htdocs" //Le répertoire qui va contenir les
                                //documents de votre site web
<Directory "C:/Apache/htdocs ">
 Options All
 AllowOverride All
 Allow from all
</Directory>
```

<Directory> et </Directory> sont utilisés pour "encapsuler" un groupe de directives applicables uniquement au réprtoire indiqué ainsi qu'àses sous-répertoires.

La directive Options contrôle quelles fonctions du serveur sont disponibles dans un répertoire particulier.

option *peut valoir **None**, auquel cas aucune fonction supplémentaire n'est disponible, ou une ou plus des possibilités suivantes :*

***All** : toutes options sauf MultiViews.*
***ExecCG**I : L'exécution des scripts CGI est autorisée.*
***FollowSymLinks** : Le serveur est autorisé àsuivre les liens symboliques dans ce répertoire.*

AllowOverride = Surcharge : Lorsque le serveur trouve un fichier .htaccess (comme spécifié par AccessFileName) il doit savoir lesquelles des directives declarées dans ce fichier peuvent outrepasser les droits fixés par des directives précédentes. *Surcharge* peut être pris comme ***None***, auquel cas le serveur ne lit pas le fichier, ***All*** auquel cas le serveur prendra en compte toutes les directives dans ce fichier

*La directive **allow** permet de configurer quels hôtes peuvent accéder àun répertoire donné.* L'*hôte* est àchoisir parmis ce qui suit :

***all** : Tous les hôtes peuvent accéder au répertoire*
***Un nom de domaine (partiel) :** Les hôtes dont les noms correspondent, ou terminent par cette chaîne sont autorisés en accès. Exemple : allow from .ncsa.uiuc.edu*

La directive ***deny*** empêche certains hôtes nommé d'accéder àun répertoire. Elle peut prendre les mêmes valeurs que celles de allow.

Installation de PhP4

1. Récupérer la dernière version de PhP (php-4_X_X-win32.zip), du site web : www.php.net.

2. Décompresser l'archive PhP dans le réépertoire c:\php\

   Les fichiers de l'archive sont : php.exe, php.ini-dist (ou php.ini), licence.txt, readme.txt, php4ts.dll, msvcrt.dll plus d'autres DLL.

3. S'il existe un fichier nommé php.ini-dist, et pas de fichier php.ini, renoomer php.ini-dist par php.ini

4. Déplacer le fichier : *php.ini* sous le répertoire windows.

5. Editer le fichier php.ini et procéder aux modifications suivantes au niveau de la section Paths and Directories :

extensions_dir=c:\php4 //la variable extensions_dir détermine le dossier dans lequel //se trouvent les DLL

include_path = . //include_path détermine les répertoires àouvrir quand le script recherche un fichier. La valeur par défaut est (.) ce qui signifie répertoire courant.

Au niveau d la section Windows Exensions, décommenter la ligne des fichiers d'extension que l'on souhaite utiliser.

Par exemple :

***php_calendar.dll** : Fonction de conversions calendaires*
***php_crypt.dll** : Fonction de cryptage*
***php_dbase.dll** : Fonctions DBase*
***php_dbm.dll** : Emulation GDBM avec la librairie Berkely DB2*
***php_filepro.dll** : Accès aux bases de données File pro (accès en lecture seule).*
***php_gd.dll** : Manipulation des images GIF/JPG/PNG avec GD Library*
***php_hyperwave.dll** : Fonctions HyperWave*
***php_imap4r2.dll** : Fonctions IMAP 4*
***php_ldap.dll** : Fonctions LDAP*
***php_msql1.dll** : Fonctions client mSQL 1*
***php_msql2.dll** : Fonctions client mSQL 2 client*
***php_mssql.dll** : Fonctions client MSSQL client (requiert MSSQL DB-Libraries)*
***php_mysql.dll** : Fonctions MySQL*
***php_nsmail.dll** : Fonctions Netscape mail*
***php_oci73.dll** : Fonctions Oracle*
***php_zlib.dll** : Fonctions ZLib*
***php_snmp.dll** : Fonctions SNMP get et walk (NT seulement!)*

6. Il faut maintenant configurer Apche pour qu'il puisse supporter PhP. Editer le fichier c:\Apache\conf\httpd.conf.

Tout d'abord il faut définir les extensions des fichiers qui seront interprétés par l'analyseur PHP, avec la directive **AddType**.

On peut par exemple définir *.php*, *.php3* et *.phtml.*

```
AddType application/x-httpd-php .php
AddType application/x-httpd-php .php3
AddType application/x-httpd-php .phtml
```

Ensuite il faut indiquer àApache d'utiliser PHP pour ces fichiers :

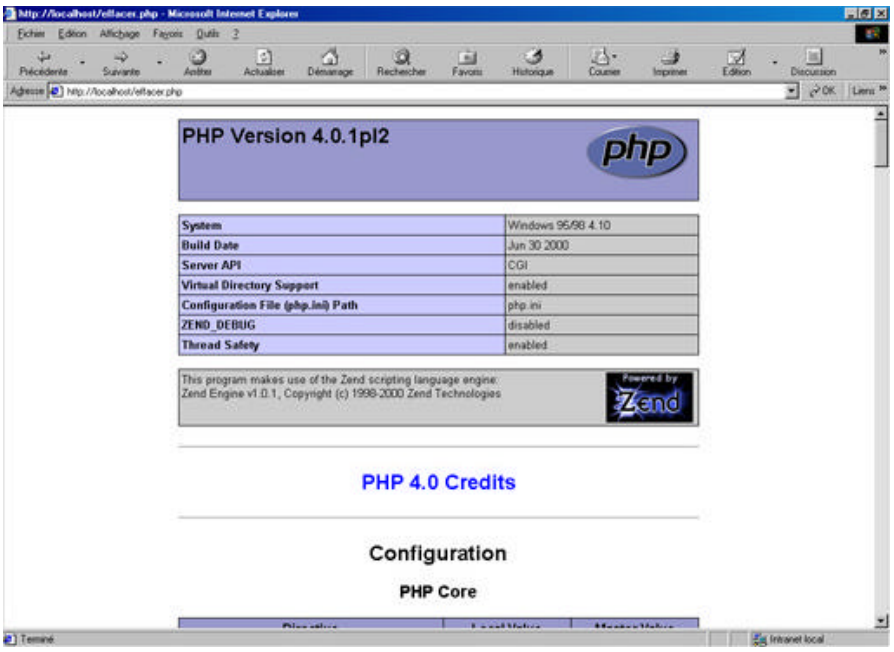
```
ScriptAlias /php/ c:/php4/
Action application/x-httpd-php /php/php.exe
```

Lorsque Apache rencontre un fichier de type *application/x-httpd-php*, il demande donc à*c:/php/php.exe* de l'analyser.

7. Test de la configuration : Pour vérifier que PhP fonctionne correctement, on crée un fichier de script : test.php, dont on insére la ligne de code suivante :

```
<? phpinfo(); ?>
```

Sauvegarder le fichier test.php sous le répertoire c:\Apache\htdocs, puis lancer ce fichier àl'aide du navigateur : http://localhost/test.php. On doit obtenir une page d'information sur la configuration de PHP :



L'interface phpMyAdmin

Présentation :

PhpMyAdmin est un ensemble de scripts PHP3 permettant de gérer des bases MySQL via une interface accessible par n'importe quel navigateur Web. Il permet entre autres de :

- créer et supprimer des bases (par l'administrateur seulement)
- créer, copier, supprimer, et modifier des tables
- supprimer, éditer et ajouter des champs
- éxecuter des requêtes SQL, voire des "batchs" SQL
- gérer les clés des tables
- importer des fichiers texte dans des tables
- créer et intégrer des dumps de tables

Pour l'installer, il suffit de décompresser l'archive dans un répertoire de notre serveur Web qui sera protégé par un *.htaccess*, de plus certains fichiers de ce répertoire auront des droits restreints.

Ensuite vous devez éditer le fichier *config.inc.php3* pour configurer votre installation de phpMyAdmin. Nous allons configurer un seul serveur (en effet, phpMyAdmin est capable d'administrer plusieurs serveurs MySQL), une authentification avancée (utilisateur/password) et la langue utilisée sera le français.

Comme vous pouvez le constater, on est obligé de coder dans ce fichier le nom et le mot de passe de l'administrateur. Nous allons protéger le fichier de configuration *config.inc.php3* par un **".htaccess"** qui interdit àtout le monde d'accéder au fichier. Créez le fichier *.htaccess* dans le répertoire d'installation de phpMyAdmin :

```
<Files config.inc.php3>
Order Deny,Allow
Deny From All
</Files>
```

Les étapes de l'installation :

Pour l'installer, il suffit de décompresser l'archive dans un répertoire de notre serveur Web qui sera protégé par un *.htaccess*, de plus certains fichiers de ce répertoire auront des droits restreints.
Les étapes de l'installation sont détaillées ci-dessous :

1. Récupérer la dernière version de PhpMyAdmin (Site officiel : http://www.phpwizard.net/projects/phpMyAdmin/)

Format tar : http://www.phpwizard.net/projects/phpMyAdmin/phpMyAdmin_2.1.0.tar.gz
Format zip : http://www.phpwizard.net/projects/phpMyAdmin/phpMyAdmin_2.1.0.zip

2. Décompresser le fichier dans un répertoire du serveur web (attention àconserver l'arborescence du fichier compressé !)

Par exemple : /htdocs/phpmyadmin

3.Ouvrir le fichier **config.inc.php3** dans votre éditeur favori et modifier les valeurs de l'**hôte**, de l'**utilisateur**, du **mot de passe** et de la **langue** pour adapter phpMyAdmin à votre environnement :

Au début du fichier :

$cfgServers[1]['host'] = '**localhost**';
$cfgServers[1]['user'] = '**utilisateur**';
$cfgServers[1]['password'] = '**mot de passe**';

La ligne **require("english.inc.php3");** définit la langue utilisée par l'interface de phpMyAdmin. Remplacer english par **french** (ou toute autre langue, àcondition que le fichier langue.inc.php3 existe).
Toutes les autres variables sont décrites dans le fichier **Documentation.html** fourni avec phpMyAdmin. Une **FAQ** y est aussi donnée dans cette page, qui contient des tas d'informations utiles. A lire impérativement.

4. Protéger éventuellement (recommandé) le répertoire dans lequel vous avez installé phpMyAdmin, par exemple en utilisant un système d'authentification HTTP-AUTH (fichier **.htaccess** pour Apache).

5. Ouvrir le fichier **<http://localhost>/<rep-install-phpmyadmin>/index.php3** dans votre navigateur. phpMyAdmin doit afficher un écran de bienvenue ainsi que vos bases de données.

Exemple du fichier config.inc.php3 :

```
<?php
/* $Id: config.inc.php3,v 1.26 1999/10/10 19:57:29 tobias Exp $ */

/*
 *  phpMyAdmin Configuration File
 *  All directives are explained in Documentation.html
 */

// The $cfgServers array starts with $cfgServers[1].  Do not use $cfgServers[0].
// You can disable a server config entry by setting host to ''.
$cfgServers[1]['host'] = 'localhost';          // L'idresse IP locale / hostname
$cfgServers[1]['port'] = '';                   // MySQL port - leave blank for default port
$cfgServers[1]['adv_auth'] = false;            // mettre false pour un test en local avec win95/98
authentication?
$cfgServers[1]['stduser'] = 'root';            // MySQL standard user (only needed with advanced auth)
$cfgServers[1]['stdpass'] = '';                // MySQL standard password (only needed with advanced
auth)
$cfgServers[1]['user'] = 'root';               // MySQL user (only needed with basic auth)
$cfgServers[1]['password'] = '';               // MySQL password (only needed with basic auth)
$cfgServers[1]['only_db'] = '';                // Le root peut voir toutes les  bases
$cfgServers[1]['verbose'] = '';                // Verbose name for this host - leave blank to show the
hostname

$cfgServers[2]['host'] = '';
$cfgServers[2]['port'] = '';
$cfgServers[2]['adv_auth'] = false;
$cfgServers[2]['stduser'] = '';
$cfgServers[2]['stdpass'] = '';
$cfgServers[2]['user'] = '';
$cfgServers[2]['password'] = '';
$cfgServers[2]['only_db'] = '';
$cfgServers[2]['verbose'] = '';

$cfgServers[3]['host'] = '';
$cfgServers[3]['port'] = '';
$cfgServers[3]['adv_auth'] = false;
$cfgServers[3]['stduser'] = '';
$cfgServers[3]['stdpass'] = '';
$cfgServers[3]['user'] = 'root';
$cfgServers[3]['password'] = '';
$cfgServers[3]['only_db'] = '';
$cfgServers[3]['verbose'] = '';

// If you have more than one server configured, you can set $cfgServerDefault
// to any one of them to autoconnect to that server when phpMyAdmin is started,
// or set it to 0 to be given a list of servers without logging in
// If you have only one server configured, $cfgServerDefault *MUST* be
// set to that server.
$cfgServerDefault = 1;                         // Default server  (0 = no default server)
$cfgServer = '';
unset($cfgServers[0]);

$cfgManualBase = 'http://www.tcx.se/Manual_chapter';

$cfgConfirm = true;
$cfgPersistentConnections = false;

$cfgBorder      = '0';
$cfgThBgcolor  = '#D3DCE3';
```

```
$cfgBgcolorOne = "#CCCCCC";
$cfgBgcolorTwo = "#DDDDDD";
$cfgMaxRows = 30;
$cfgMaxInputsize = "300px";
$cfgOrder = "ASC";
$cfgShowBlob = true;
$cfgShowSQL = true;

require("french.inc.php3"); //Pour avoir les pages générées en français

$cfgColumnTypes = array(
   "TINYINT",
   "SMALLINT",
   "MEDIUMINT",
   "INT",
   "BIGINT",
   "FLOAT",
   "DOUBLE",
   "DECIMAL",
   "DATE",
   "DATETIME",
   "TIMESTAMP",
   "TIME",
   "YEAR",
   "CHAR",
   "VARCHAR",
   "TINYBLOB",
   "TINYTEXT",
   "TEXT",
   "BLOB",
   "MEDIUMBLOB",
   "MEDIUMTEXT",
   "LONGBLOB",
   "LONGTEXT",
   "ENUM",
   "SET");

$cfgFunctions = array(
   "ASCII",
   "CHAR",
   "SOUNDEX",
   "CURDATE",
   "CURTIME",
   "FROM_DAYS",
   "FROM_UNIXTIME",
   "NOW",
   "PASSWORD",
   "PERIOD_ADD",
   "PERIOD_DIFF",
   "TO_DAYS",
   "USER",
   "WEEKDAY",
   "RAND");

$cfgAttributeTypes = array(
   "",
   "BINARY",
   "UNSIGNED",
   "UNSIGNED ZEROFILL");
      ?>
```

Ressources

Documentation :

Officiel

MySQL - Une des bases données les plus utilisés dans le monde de l'Internet, de nombreux hébergeurs l'utilisent. en plus c'est du GPL ...
http://www.mysql.com

Apache - Le serveur le plus populaire plus de 50% des serveurs dans le monde
http://www.apache.org

PHP.NET - Le site officiel de PHP.
http://www.php.net

Php français

PHP France - Site français sur PHP documentation, tutorials chat et forums sont au rendez vous !
http://www.phpfrance.net

PHP INFO - Site d'information sur le langage de script PHP et le serveur de bases de données avec lequel il se marie le mieux : MySQL
http://www.phpinfo.net

PHP INDEX - Site d'information sur PHP : news, forums, articles... un site très complet
http://www.phpindex.com

La communauté PHP Francophone - La communauté PHP Francophone : annuaire des developpeurs, stats
http://www.communautephp.com

Php anglais

PHP BUILDER - Un site composé essentielement d'article très complet.
http://www.phpbuilder.com

Weber dev - Beaucoups d'exemple PHP de trucs et astuces mais aussi du javascript, asp, mysql ...
http://www.weberdev.com/

HOTSCRIPTS - Une mine d'or de script PHP et autres !
http://www.hotscripts.com

Autres

ALL HTML - Un site français sur tout ce qui peut toucher le Web : HTML, DHTML, JAVASCRIPT....
http://www.allhtml.com

Outils et solutions

Editeurs :

On peut citer,

Sous Windows,

- UltraEdit : http://www.ultraedit.com/downloads/index.html
- PHPEd : http://www.soysal.com/PHPEd/
- HtmlTool : http://freedani.free.fr/htmltool.php3
- EditPlus : http://www.editplus.com/
- HTML Kit : http://www.chami.com/html-kit/
- TextPad : http://www.textpad.com/
- PHPCoder : http://www.phpide.de/
- HomeSite : http://www.allaire.com/Products/HomeSite/
- PHP Script Editor : http://www.pc-service-boerner.de/PHPScriptEditor.htm
- phpEdit : http://www.phpedit.com/
- 1st Page 2000 : http://www.evrsoft.com/1stpage/
- CuteHTML : http://www.globalscape.com/products/cutehtml/
- DreamWeaver : http://www.macromedia.com/software/dreamweaver/
- Editpad : http://www.editpadpro.net/editpadclassic.html
- GWD : http://www.gwdsoft.com/
- PrimalSCRIPT : http://www.sapien.com/PrimalSCRIPT.htm
- Jed : http://space.mit.edu/~davis/jed.html
- FTE (Folding Text Editor) : http://www.kiss.uni-lj.si/~k4fr0235/fte/
- VIM : http://www.vim.org/
- Emacs : http://www.gnu.org/software/emacs/
- NEdit : http://www.nedit.org/
- PHP mode for Emacs http://sourceforge.net/projects/php-mode/
- ConTEXT : http://www.fixedsys.com/context/

Sous Unix,

- VIM : http://www.vim.org/
- Code Crusader : http://www.coffeecup.com/select/editor.html
- Quanta+ : http://sourceforge.net/projects/quanta/
- asWedit : http://www.advasoft.com/
- BlueFish : http://bluefish.openoffice.nl/

- Jed : http://space.mit.edu/~davis/jed.html

- FTE (Folding Text Editor) : http://www.kiss.uni-lj.si/~k4fr0235/fte/

- NEdit : http://www.nedit.org/

- PHP mode for Emacs http://sourceforge.net/projects/php-mode/

Sous Mac OSX,

- BBEdit : http://www.barebones.com/products/bbedit.html

- VIM : http://www.vim.org/

Frontal SGBD

phpMyAdmin : http://phpwizard.net/projects/phpMyAdmin/index.html

phpPgAdmin : http://www.greatbridge.org/project/phppgadmin/projdisplay.php

phpOracleAdmin : http://phporacleadmin.org/

phpSybaseAdmin : http://www.itlab.musc.edu/phpSybaseAdmin/Documentation.html

IDE

Zend IDE : http://www.zend.com/store/products/zend-ide.php

KPHPDevelop : http://www.kphpdev.com/

Cache

Zend Cache : http://www.zend.com/store/products/zend-cache.php

AfterBURNER*Cache : http://bwcache.bware.it/cache.htm

APC Alternative PHP Cache : http://apc.communityconnect.com/

phpCache : http://0x00.org/phpCache/

Optimiseur

Zend Optimzer : http://www.zend.com/store/products/zend-optimizer.php

Encodeur

Zend Encoder : http://www.zend.com/store/products/zend-encoder.php

PHP Bytecode Compiler : http://pbc.sourceforge.net/

PHPCompiler : http://www.deskcode.com/phpcompiler/

Debogueur

Zend IDE : http://www.zend.com/store/products/zend-ide.php

DBG PHP Debugger : http://dd.cron.ru/dbg/

Sessions

PHPLib : http://phplib.netuse.de/

BDSM : http://www.lazycat.org/BDSM/

dbSessions : http://www.phpheaven.net/resources/libraries/dbSessions/

PostgreSQL Session Handler : http://www.csh.rit.edu/~jon/projects/php/pgsql_session_handler/

sqlsession : http://ecom1.onestop.net/~joshm/sqlsession.html

Abstraction SGBD

PHPLib : http://phplib.netuse.de/

dbMysql : http://www.phpheaven.net/resources/libraries/dbMysql/

PDAC : http://www.dsnet.net/~cris/dac.html

PHP-MySQL-Lib : http://www.claws-and-paws.com/software/

Commerce

- I-Store : http://www.w3-concept.net/

- FishCartSQL : http://www.fishcart.org/

Webmail

Akasha 1.3 : http://sourceforge.net/projects/akasha/

BasiliX : http://www.basilix.org/

DragonFlyMail : http://www.dflytech.com/products/dragonflymail/

IMP : http://www.horde.org/imp/

JAWmail : http://jawmail.sourceforge.net/

kenstermail : http://sourceforge.net/projects/kenstermail/

LearnLoop : http://www.learnloop.org/

Netmania : http://www.netmania.com

MyPhPim : http://sourceforge.net/projects/myphpim/

NOCC : http://nocc.sourceforge.net/

oMail-admin : http://omail.omnis.ch

Phorecast : http://ciaweb.net/free/phorecast/

PHPwebmail : http://www.bmpsystems.com/downloads/linux/webmail-2.1.tgz

Postaci : http://www.trlinux.com/

TMail : http://www.groovin.net/tmail/

VAMP Webmail : http://undead.thegraveyard.org/?topic=vamp

vpopmail : http://www.inter7.com/vpopmail/

werkmail : http://werkmail.sourceforge.net/

Hébergement

Il faut en premier lieu distinguer les hébergeurs gratuits et les hébergeurs professionnels. Et pour ces derniers, dissocier les offres d'hébergement en co-hosting de ceux dédiés.

Notons aussi que si certains hébergeurs proposent PHP, il n'en va pas toujours de même pour MySQL ou tout autres SGBD. C'est en particulier le cas des

principaux hébergeurs gratuits. De plus, ces derniers imposent parfois la présence de bandeaux publicitaires.

Enfin, distinguer les hébergeurs proposant PHP3 et ceux proposant PHP4.

Voici une liste non exhaustive d'hébergeurs gratuits :

- Free : http://www.free.fr

- Forez : http://www.forez.com

- Chez : http://www.chez.com

- Multimania : http://www.multimania.com

- Hébergement Gratuit : http://www.hebergement-gratuit.com

Voici une liste non exhaustive d'hébergeurs professionnels/co-hosting :

- Online : http://www.online.fr

- Nexen : http://www.nexen.fr

- OVH : http://www.ovh.com

- Amen : http://www.amen.fr

- Claranet : http://www.claranet.fr

Voici une liste non exhaustive d'hébergeurs professionnels/dédiés :

- Agarik : http://www.agarik.com

- Claranet : http://www.claranet.fr

Pour une recherche et une sélection plus fine, vous pouvez aussi consulter le site abc-hébergement sur http://www.abchebergement.com/

Scripts et applications

Où trouver des scripts et applications en PHP

- Zend : http://www.zend.com/apps.php

- HotScripts : http://www.hotscripts.com/PHP/

- PHPScripts : http://phpscripts.free.fr/

- Freshmeat : http://freshmeat.net/browse/183/

- PHPIndex : http://www.phpindex.com/annuaire/annuaire_total.php3?categorie=500

Forum

- W-Agora : http://w-agora.araxe.fr/

- Phorum : http://www.phorum.org/

- phpBB : http://www.phpbb.com/

-HotScripts : http://www.hotscripts.com/PHP/Scripts_and_Programs/Discussion_Boards/

-PHPScripts : http://phpscripts.free.fr/?page=scripts&cat=forums

-PHPIndex : http://www.phpindex.com/annuaire/annuaire_partiel.php3?sous_categorie=502&categorie=500

Chat

phpMyChat : http://www.phpheaven.net/

ChatZoneSuivante : http://www.zonesuivante.com/

phpLittleChat : http://www.gamate.com/

HotScripts : http://www.hotscripts.com/PHP/Scripts_and_Programs/Chat_Scripts/

PHPScripts : http://phpscripts.free.fr/?page=scripts&cat=chats

Vote

- SimplePoll : http://www.quake-au.net/php/

- phpAnyVote : http://www.punknews.org/code/

- phpVault : http://www.phpvault.com/

-HotScripts : http://www.hotscripts.com/PHP/Scripts_and_Programs/Polls_and_Voting/

-PHPScripts : http://phpscripts.free.fr/?page=scripts&cat=polls

-PHPIndex : http://www.phpindex.com/annuaire/annuaire_partiel.php3?sous_categorie=511&categorie=500

Guestbook

Probook : http://www.devpros.com/

phpBook : http://www.netone.at/

phpMyGB : http://narcolepsy.dhs.org/mysoftware/

HotScripts : http://www.hotscripts.com/PHP/Scripts_and_Programs/Guestbooks/

PHPScripts : http://phpscripts.free.fr/?page=scripts&cat=livresdor

Portail

phpNuke : http://www.phpnuke.org/

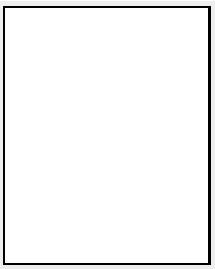
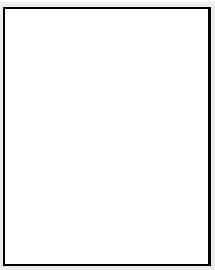
Drupal : http://www.drop.org/

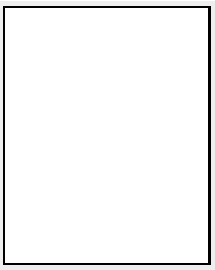
ThatWare : http://www.thatware.org/

HotScripts : http://www.hotscripts.com/PHP/Scripts_and_Programs/Portal_Systems/

PHPScripts : http://phpscripts.free.fr/?page=scripts&cat=portails

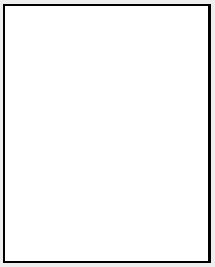
Livres :

	PhP Professionel
	De : **J. Castagnetto, B. Moon, H. Rawat, S. Schumann, C. Scollo ...** Editeur : **Wrox Press** Paru en : **octobre 2000** Nombre de pages : **954** Prix éditeur : **340 F**
	Programmation web avec php
	De : **L. Lacroix, N. Leprince, C. Boggero, C. Lauer** Editeur : **Editions Eyrolles** Paru en : **avril 2000** Nombre de pages : **364** Prix éditeur : **230 F**
	Programmation en php



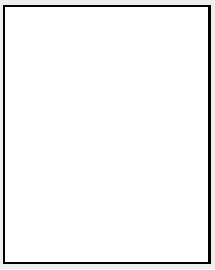
De : **Leon Atkinson**
Editeur : **Campus Press**
Paru en : **mai 2000**
Nombre de pages : **540**
Prix éditeur : **229 F**

Grand Livre PHP4 & MySQL



De : **G. A. Leierer, R. Stoll**
Editeur : **Micro Applications**
Paru en : **août 2000**
Nombre de pages : **470**
Prix éditeur : **209 F**

Pages Web dynamiques avec ASP, PHP, SQL



De : **Herellier, Mérigod**
Editeur : **Campus Press**
Paru en : **décembre 1999**
Nombre de pages : **326**
Prix éditeur : **135 F**