SOMMAIRE

Introduction 2

Présentation du travail réalisé 4
Chapitre N°1 : Les Bases 4
Chapitre N°2 : Les Variables 5
Chapitre N°3 : Les Opérateurs 14
3-1 Les opérateurs numériques
3-2 Les opérateurs de comparaisons 15
3-3 Les opérateurs logiques
3-4 Les opérateurs « caractères »
Chapitre N°4 : Les Conditions et Boucles 16
4-1 Les expressions conditionnelles 17
4-2 Les boucles 19
Chapitre N°5 : Les Fichiers 24
5-1 Fonctions tests
5-2 Ouverture et fermeture de fichiers 26
5-3 Lecture de données 27
5-4 Écriture de données
5-5 Autres fonctions utiles 28
5-6 Importation de fichiers Php 29
Chapitre N°6 : PHP et MySQL 30
6-1 Connexion et déconnexion à la base de données
6-2 État de MySQL 31
6-3 État d'une base 32
6-4 Requêtes sur les tables 34
Chapitre N°7 : Les Bases de données 36
7-1 Accéder aux bases de données avec Php
7-2 Se connecter à une base de données MySQL 37
Chapitre N°8 : Les Requêtes sur les bases de données 38
8-1 Fonctionnement : cas de la clause SELECT
8-2 Les principales fonctions MySQL en Php 40
Chapitre N°9 : Autres requêtes sur les bases de données 42

Conclusion

Bibliographie

Introduction

PHP - MYSQL

Ce cours est une prise en main du langage PHP (Version 4), du langage SQL via le logiciel de gestion de bases de données MySQL (Version 3.23) ainsi que l'interfaçage de MySQL via des scripts PHP.
Pour suivre ce cours vous devez déjà avoir des notions d'algorithmiques, de HTML et quelques notions sur les serveurs web pour l'installation et l'hébergement de pages web (html et CGI).

Introduction

PHP qui signifie Personal Home Page, est apparu en 1994, sous forme de petits outils pour faciliter la vie des programmeurs web notamment grâce à Rasmus Lerdorf.

Pour quelle raison utiliser et développer en PHP alors qu'il existe une multitude (voir plus :-)) de langages de programmation (PERL, C, java ...) ?
PHP est un langage de scripts. Les versions antérieures (1, 2 et 3) étaient interprétées, par conséquent il ne nécessitaient pas d'être compilés pour obtenir un objet, un exécutable avant d'être utilisable (comme en C par exemple).
Mais aujourd'hui, depuis la Version 4, le PHP est un langage compilé.

PHP est un module supporté par le serveur web Apache, le plus répandu dans le monde (plus de 70% des serveurs web), il est donc développé pour être facilement utilisable via ce serveur (Il fonctionne évidemment avec d'autres serveurs web).
PHP permet d'interfacer très facilement de très nombreuses bases de données notamment MySQL gratuite et performante.
On retrouve d'ailleurs l'ensemble Apache-PHP-MySQL souvent sur la plate-forme web. Il offre ainsi des outils et de nombreuses fonctions facilitant ce travail.

Du fait de l'utilisation par un grand nombre de ce langage vous avez accès sur le net et dans la littérature à de nombreuses sources d'informations et d'aides (Tutoriel, forum de discussions, ...).

Un des gros avantages de PHP sur d'autres langages comme PERL est l'intégration dans la même page du code HTML « brut » et du code PHP. Plus besoin de réaliser une page HTML et une deuxième dans le langage de programmation désiré, ou de faire une page dans laquelle le code HTML est « en capsulé » dans le code du langage de programmation de manière plus ou moins simple. Vous pouvez avec PHP taper vos lignes de codes en HTML, puis intégrer ou vous le désirez du code PHP et ainsi de suite. Travailler de cette manière et sur un seul fichier vous fait gagner énormément de temps et de clarté dans votre site.
Exemple, vous réalisez un formulaire en HTML pour une enquête :
Si vous faites un fichier HTML puis un autre pour le traitement des données soumises par l'utilisateur, vous allez jongler entre ces deux fichiers pour retrouver les variables utilisées, les valeurs affectées pour le traitement dans votre CGI. Avec PHP tout est dans le même, la lisibilité sera beaucoup plus grande.

Autre avantage énorme, les scripts PHP n'ont pas besoin d'être mis dans des répertoires exécutables sur votre site comme peuvent l'être les programmes PERL, C ... Vous pouvez les inclure n'importe où. Ceci est terriblement avantageux surtout si vous n'êtes pas maître de la configuration de votre serveur web (Partie ScriptAlias dans la configuration de

httpd.conf d'Apache). Grâce à cette facilité, vous pourrez la plupart du temps faire de la programmation chez votre provider qui généralement ne vous permet pas de la réaliser à travers des répertoires exécutables traditionnels.

PHP fonctionne si l'hébergeur possède le moteur PHP. En effet le fonctionnement est le suivant : une page contenant du code PHP est appelée, il passe alors d'abord par le Préprocesseur PHP qui transforme le code PHP en code HTML et envoie ensuite seulement la page à l'utilisateur. A aucun moment le code PHP ne pourra être vu. Ce qui sécurise vos scripts et vos sources de programmes.

Un dernier élément en faveur de PHP n'est pas des moindres, sa portabilité. Si vous développez un programme PHP, vous pourrez le porter sur toutes les machines sans avoir la nécessité de modifier le code source, il suffit que le serveur web soit configuré de manière correcte et que PHP soit sur la machine.

Présentation du travail

Chapitre N°1 : Les Bases

Les bases du PHP

Les fichiers contenant des scripts PHP doivent posséder l'extension PHP. Attention la version actuelle de PHP est 4, si vous utilisez encore la version 3 vos scripts devront avoir l'extension php3.
Vous devez aussi faire attention dans ce dernier cas à la configuration de votre serveur apache. Par défaut il n'accepte pas cette extension. Vous devez dé commenter les lignes contenant le mot clé Addtype en rapport avec php3.

Lorsque vous incluez du code PHP dans un script vous devez indiquer que cette partie à interpréter est dans ce langage comme si vous incluez du Java Script par exemple. Plusieurs balises sont utilisables pour indiquer le début et la fin de la partie en langage PHP.

Vous pouvez utiliser l'une des trois suivantes :

```
<?php
codes                    php                    ...
?>
```

```
<?
codes                    php                    ...
?>
```

```
<script                      language="php"
codes                    php                    ...
?>
```

- Dans les trois cas la première ligne correspond à l'ouverture de la partie contenant le script en php.
- La deuxième ligne simule le code php.
- La troisième ligne indique la fin de la partie contenant le code php.
- Avant et après vous pouvez insérer du code HTML brut.

Chaque instruction de code php doit être terminée par un point virgule.

Les commentaires très importants dans vos programmes doivent être intégrés de la manière suivante :
Si votre commentaire ne se situe que sur une ligne vous pouvez le faire précéder par deux slashs // ou par un dièse #. Par contre si ce commentaire se situe sur plusieurs lignes utilisez alors la même syntaxe qu'en C /* pour l'ouverture */ pour la fermeture.

Exemple d'une page HTML contenant du code php 4

```
<HTML>
<HEAD>
<TITLE>
Première page php
</TITLE>
</HEAD>
<BODY BCOLOR=#FFFFFF>
<FONT SIZE=6>
<?php
print("Page d'index");
print("<BR>Saut de page HTML dans le script");
?>
</FONT>
</BODY>
</HTML>
```

Votre serveur qui possède l'interpréteur php a d'abord « traduit » la partie script en HTML. Ceci est fait en local donc sur votre propre machine et non sur la machine du client. Ensuite le code HTML est transmis. A aucun moment le client ne peut voir la partie code php, ce qui représente un avantage non négligeable.

Chapitre N°2 : Les Variables

2-1 Les variables de base

Tout comme PERL, le type de la variable dépend de ce que vous mettez dedans et de ce que vous en faites. Par contre vous pouvez forcer le type d'une variable ou modifier ce type si nécessaire.
Le caractère qui précède toutes les variables php est le dollar : $.

```
$variable = "chaine";
$variable = 3;
```

2-1-1 Les variables dynamiques

Vous pouvez créer des variables dynamiques, le nom de la variable est lui même une variable. La notation est le double dollar : $$.

```
$variable = "etudiant";
$$variable = "pat";
```

L'évaluation de la variable $etudiant donnera la valeur pat. Cette possibilité permet ainsi de créer des variables dont le nom se modifie grâce au programme que vous faites: etudiant0, etudiant1 ... Pour l'affichage vous pouvez par exemple utiliser la syntaxe :

```
print $$variable;
```

Ou

```
print "La valeur de $variable est : ${$variable}";
```

Exemple d'une page utilisant des variables dynamiques

```
1                                                    <HTML>
2              <BODY                BGCOLOR=#FFFFFF>
3                                                      <B>
4  <U>Petit script pour l'affichage de variable dynamique</U>
5                                                  <BR><BR>
6                <FONT                  COLOR="green">
7                                                    <?php
8          $variable          =          "etudiant";
9            $$variable             =            "pat";
10  print "Syntaxe 1 : La valeur de $variable est : ".$
$variable;
11                                                     ?>
12                                                   <BR>
13                                                   <?php
14  print "Syntaxe 2 : La valeur de $variable est : $
{$variable}";
15                                                     ?>
16                                                 </FONT>
17                                                 <BR><BR>
18            Fin            de            l'affichage
19                                                   </BODY>
20 </HTML>
```

Explications

- De la ligne 1 à la ligne 6 le code est du code HTML simple.
- La ligne 7 permet d'insérer du code PHP
- La ligne 8 est une affectation de la valeur etudiant dans la variable variable
- La ligne 9 introduit une variable dynamique $$variable qui est équivalent à $etudiant
- La ligne 10 et la ligne 14 propose deux méthodes pour l'affichage du contenu de la variable dynamique.

Utilisation des références dans les variables

Vous pouvez travailler sur les références (pointeur) comme en C par exemple. Ceci vous permettra par exemple de conserver la persistance d'une valeur à l'intérieur de variables et de la modifier quelque soit sa position dans le programme. Vous devez préfixer votre variable par le caractère : &

$pointeur = &$variable;

La variable pointeur ne représentera pas le contenu mais la référence qui pointe sur la valeur

de la variable variable (la zone mémoire où elle est stockée). Si la valeur de la variable est modifiée et ceci quelque soit l'endroit où elle se situe dans la programme, la valeur de la variable pointeur le sera aussi.

Exemple

```
1                                                    <HTML>
2              <BODY              BGCOLOR=#FFFFFF>
3                                                      <B>
4  <U>Travail sur les références des variables</U>
5                                                  <BR><BR>
6                <FONT                COLOR="green">
7                                                    <?php
8            $variable            =            "pat";
9                              $pointeur=&$variable;
10  print "Etape 1 : \$pointeur pointe sur la valeur
->$pointeur";
11                                                   ?>
12                                                 <BR>
13                                                 <?php
14  print "Je modifie la valeur de la variable \
$variable<BR>\n";
15                             $variable="ibulaire";
16  print "Etape 2 : ".'$pointeur'." pointe sur la valeur
->$pointeur\n";
17                                                   ?>
18                                               </FONT>
19                                               <BR><BR>
20            Fin           de           l'affichage
21                                                 </BODY>
22 </HTML>
```

Explications

- De la ligne 1 à la ligne 6 le code est du code HTML simple.
- La ligne 7 permet d'insérer du code PHP
- La ligne 8 est une affectation de la valeur pat dans la variable $variable
- La ligne 9 permet de ranger dans la variable $pointeur non pas la valeur de la variable $variable mais la référence mémoire de cette variable grâce à l'utilisation du caractère &
- La ligne 10 permet d'afficher le contenu de la variable $pointeur. Notez que pour permettre d'afficher à l'écran $pointeur et non la valeur de cette variable je désactive le caractère spécial $ au moyen du caractère \. Même chose à la ligne 14. La ligne 16 montre un autre moyen d'obtenir le même résultat grâce à l'utilisation des caractères ' et le caractère de concaténation. . J'insère dans le print un retour chariot à la fin de la ligne (\n) simplement pour que le source soit mieux présenté.
- La ligne 15 modifie la valeur de la variable $variable.

- La ligne 16 permet de se rendre compte que malgré le fait que nous n'intervenons pas directement sur la variable $pointeur, celle ci voit sa valeur modifiée, c'est « l'effet pointeur ».

Si vous n'aviez pas affecté à la variable $pointeur la référence vous auriez alors obtenu le résultat suivant : modification de la ligne 10 en :

10 *$pointeur=$variable;*

Quelques fonctions de traitement des chaînes

- **strlen**
  Cette fonction va permettre de trouver le nombre de caractères contenus dans une chaîne simple.

```
$nbcar=strlen($variable);
```

- **strstr**
  Renvoie VRAI si la sous chaîne existe dans la chaîne. Attention : pour la fonction *strstr* la casse est importante.

```
strstr($chaine,$souchaine);
```

- **substr**
  Permet de récupérer une partie de la chaîne en indiquant l'index du caractère à partir duquel la recherche se fait. Vous pouvez aussi indiquer le nombre de caractère à récupérer. Si l'index est négatif, la recherche se fait depuis la fin de la chaîne.

```
$recup=substr($variable,$index,$longueur);
```

- **strtok**
  Permet de récupérer les éléments séparés par un ou des délimiteurs. Attention, le délimiteur n'apparaît pas dans la chaîne récupérée. Lorsque vous utilisez cette fonction, la chaîne de départ est stockée en mémoire, ce qui permet ensuite de pouvoir relancer strtok sans nécessité d'indiquer le nom de la variable où se passe la recherche. Ce qui est pratique sinon vous referiez la recherche toujours sur la même partie et obtiendriez toujours le même résultat à savoir la première occurrence trouvée. Vous pouvez donc faire la recherche sur toute la chaîne en utilisant une boucle *while*.

```
$recup=strtok($variable,$delimiteur);
```

  Ou

```
$recup=strtok($delimiteur);
```

- **explode**
  Cette fonction renvoie dans un tableau, les valeurs comprises entre un séparateur

que vous indiquez en argument. A la différence de *strtok*, même si vous indiquez plusieurs caractères comme délimiteurs, celui-ci ne sera vu que comme un seul.

```
$liste=explode($delimiteur,$variable);
```

- **implode**
  Cette fonction a l'effet inverse de explode. Elle permet à partir d'une liste de créer une variable simple. Les différents éléments de la liste seront « recollés » via ce séparateur.

```
$variable=implode($liste,$delemiteur);
```

- **split**
  Cette fonction permet de séparer les éléments d'une variable simple en utilisant des expressions régulières (et, ou ...). Attention la casse de l'expression régulière est très importante (nombre de blancs dans le séparateur ...). Le résultat est stocké dans une liste.

```
$liste=split("$delimiteur",$variable);
```

- **trim, ltrim et chop**
  Ces trois fonctions permettent de supprimer à différents endroits d'une chaîne simple, les caractères « neutres » tels les espaces, tabulations, retour à la ligne ... Ces fonctions renvoient une chaîne « nettoyée ». *trim* les supprime tous en début en et fin de chaîne. *ltrim* les supprime tous en début de chaîne. *chop* les supprime tous en fin de chaîne.

```
$variable=trim($variable);
$variable=ltrim($variable);
$variable=chop($variable);
```

- **str_replace**
  Cette fonction remplace toutes les occurrences d'une chaîne par une autre.

```
$nettoie=str_replace($avant,$apres,$variable);
```

- **strtoupper et strtolower**
  Ces fonctions permettent de transformer en majuscule ou minuscule tous les caractères contenus dans une chaîne.

```
$nouveau=strtoupper($variable);
$nouveau=strtolower($variable);
```

- **ucwords** Cette fonction permet de mettre en majuscule, tous les premiers caractères de chaque mot contenu dans une chaîne simple.

```
$modif=ucwords($variable);
```

- **addslashes et stripslashes**
  Addslashes permet de mettre un slashe devant les caractères apostrophe ('), double apostrophe (") et le slashe (\) lui-même. Cette fonction peut être très utile lors de requête SQL. Ces caractères étant spécifiques, les décommenter devient important. Stripslashes est la fonction inverse.

```
$modif=addslashes($variable);
$modif=stripslashes($variable);
```

2-1-2Les variables composées

Les listes

Les listes sont des tableaux à deux dimensions. La première est un indice numérique entier et la deuxième la valeur associée à cet indice. Une liste permet ainsi de stocker plus d'un valeur dans un même structure. Imaginez un meuble constitué de rangements. Une colonne servira à indiquer l'ordre de rangement on appelle aussi cela une clé (ici les indices numériques) tandis que l'autre colonne servira à ranger les valeurs (un objet). Vous pouvez accéder à un objet directement en indiquant le numéro de la case où il est rangé.

Pour définir une liste vous utiliserez le même caractère que pour une variable simple : $
L'affectation se fera de la façon suivante :

1. Affectation complète de la liste

```
$liste=array("escargot de bourgogne","cagouille","manchot","bretagne","php4");
```

2. Affectation d'une valeur à un indice

```
$liste[0] = escargot de bourgogne";
```

3. Indiçage automatique
   php4 permet de réaliser une incrémentation automatique des indices sans avoir à s'en soucier. Vous ajoutez une valeur qui sera affectée automatiquement au dernier indice + 1.
   La syntaxe est la suivante :

```
$liste[]="nouvelle valeur";
```

Pour la dernière syntaxe, nous n'indiquons pas de valeur à l'indice. Ce système peut être très utile dans les formulaires où vous donnerez comme nom de l'identifiant pour des boutons de type checkbox, des listes type select ... une liste auto incrémentée.

```
$liste=array("escargot de bourgogne","cagouille","manchot","bretagne","php4");
```

$liste[1] renverra la valeur cagouille

Attention les indices commencent à la valeur 0.

Vous pouvez utiliser la fonction *count* qui vous renverra le nombre d'élément contenu dans la liste.

Les tableaux associatifs à deux dimensions

Ces tableaux, à la différence des listes, auront comme indice des chaînes de caractères. Ces indices se nomment des clés. Une clé est forcément associée à une valeur. Vous obtenez donc un couple.
Pour définir un tableau vous utiliserez le même caractère que pour une variable simple ou pour une liste : $
L'affectation se fera de la façon suivante :

```
$tableau=array(nom=>Jacquenod,prenom=>frederic,age=>"En evolution constante");
```

Si la clé ou la valeur n'est pas un simple mot ou comporte des caractères blancs ... n'oubliez pas les guillemets.

Affectation d'une valeur à un indice:

```
$tableau[adresse]="Paris 75";
```

Pour parcourir cette structure vous utiliserez la boucle *foreach* (voir boucles) .

Il existe un tableau associatif prédéfini qui se nomme GLOBALS et qui contient les informations sur le script, son contenu (variables ...) et le serveur web.
D'autres tableaux associatifs existent comme HTTP_GET_VARS et HTTP_POST_VARS qui contiennent les paires nom-valeurs en fonction de la méthode de transmission utilisée dans le formulaire (get ou post). Ces deux tableaux peuvent être utiles lorsque vous ne savez pas quels sont les variables qui seront renvoyées par le formulaire.

Les tableaux associatifs multidimensionnels

Le tableau associatif HTTP_POST_VARS est un tableau particulier car multidimensionnel.
Pour le moment nous n'avons vu que les tableaux à 2 dimensions.
Mais rien n'empêche d'associer un autre tableau à un élément du premier. Vous aurez un tableau de tableaux.
Imaginons un tableau contenant les informations sur chaque cours donné dans une université. Un cours est caractérisé par son intitulé, son professeur, la durée du cours, le lieu ...

Affectation complète du tableau associatif

```
$tableau=array(
array(intitule=>reseau,professeur=>"Jacquenod
Frédéric",duree=>4,lieu=>paris),
array(intitule=>unix,professeur=>"Petit
Jean",duree=>2,lieu=>jussieu),
array(intitule=>biologie,professeur=>"Mobby
Dick",duree=>5,lieu=>sorbonne)
);
```

Chaque élément du tableau est lui-même un tableau. Chaque tableau est rangé dans une case du tableau global dont la référence est un indice numérique (nous avons plutôt à faire à une liste de tableau).
Le premier tableau est situé à l'indice 0 et ainsi de suite. Ceci du fait que nous n'avons spécifié aucune clé. Mais rien n'empêche d'en spécifier une.

```
$tableau=array(
cours1=>array(intitule=>reseau,professeur=>"Jacquenod
Frédéric",duree=>4,lieu=>paris),
cours2=>array(intitule=>unix,professeur=>"Petit
Jean",duree=>2,lieu=>jussieu),
cours3=>array(intitule=>biologie,professeur=>"Mobby
Dick",duree=>5,lieu=>sorbonne)
);
```

Affectation d'une valeur à un indice

```
$tableau[0][duree]=5;
```

ou

```
$tableau[cours1][duree]=5;
```

Dans les deux cas on affecte la valeur 5 à la clé duree du premier tableau.

Travail sur le type de la variable

Par défaut, PHP ne réclame pas de typage des variables, c'est à dire la déclaration de leur type, cependant il est possible de "typer" et de connaître le type d'une variable Parfois cela pourrait pourtant être utile. Voici les fonctions qui le permettent.

gettype et settype

PHP4 permet grâce aux fonctions *gettype* et *settype* d'obtenir le type d'une variable ou de modifier ce type.

```
gettype($variable);
```

et

```
settype ($variable,string);
```

Type renvoyé par gettype	Type de la donnée
Integer	entière
double	réelle
string	chaîne simple
array	liste ou tableau
object	Objet (pointeur)
class	classe
NULL	Type inconnu

Tester le type de la variable

Vous pouvez utiliser des fonctions permettant de tester le type de la variable et elles renvoient VRAI (ou 1) si le type est le bon.

- *is_int()*, *is_integer()*, *is_long()* renverront VRAI si le type est entier
- *is_double()*, *is_float()*, *is_real()* renverront VRAI si le type est double (réel)
- *is_string()*, *is_array()*, *is_object()* renverront respectivement VRAI si le type est string, array ou object

Typage des variables

Vous pouvez aussi forcer le type de la valeur en utilisant les fonctions *intval()*, *doubleval() et strval()*.
Exemple

```
1                                                    <?
2    $variable    =    "3    petits    cochons";
3                      if(is_string($variable))
4                                                    {
5   print   "La   variable   est   un   :   <FONT
COLOR=green>".gettype($variable);
6              print                "</FONT><Br>\n";
7  print "Je la transforme en un entier<Br>\n";
8        $trans        =        intval($variable);
9   print   "La   variable   est   un   :   <FONT
COLOR=green>".gettype($trans);
10             print                "</FONT><Br>\n";
11  print "Sa valeur est maintenant : <FONT
COLOR=green>$trans</FONT>";
12                                                   }
13                                                 else
14  print "La variable est de type : <FONT
COLOR=green>".gettype($variable);
15 ?>
```

Les constantes

Vous disposez aussi de la possibilité de créer des constantes qui auront donc la même valeur tout au long de votre programme. Ceci peut etre utile par exemple pour positionner une valeur de débugage DEBUG qui tracera les étapes du programme si elle vaut 1 et ne fera rien si elle vaut 0 :

```
define ("DEBUG",1);
```

Vous pouvez aussi vous en servir pour positionner un préfixe pour vos pages web que vous intégrerez dans vos liens hypertexte, vos références d'images ...

```
define("PRE","http://www.netalya.com");
```

On l'appelle ensuite comme ceci :

```
<php echo PRE ?>
```

Tests sur les variables

Vous avez à votre disposition les fonctions *isset*, *unset* et *empty* qui vous permettent de tester la validité, l'existence d'un contenu ...

- *isset* permet de tester si la variable existe
  if(isset($variable)){print "La variable est définie et contient une valeur";}
  renverra VRAI si la variable contient une valeur
- *unset* libère la place mémoire réservée par la variable
  unset($variable);
  Une fois cette action réalisée, le test avec isset renvoie FAUX
- *empty* renvoie VRAI si la variable n'est pas définie ou si elle renvoie une valeur FAUSSE (valeur 0 ou chaîne vide)
  if(empty($variable)){print "variable non définie ou contient 0 ou chaine vide";}

Chapitre N°3 : Les Opérateurs

3-1 Les opérateurs numériques

Ils s'appliquent à des opérations ayant comme résultat des valeurs de types numériques (integer, double).

Pour les exemples donnés ci-dessous on utilise les variables initialisées de la manière suivante :

```
$op1=5;
$op2=2;
```

Opérateur	Signification	Exemple	-> (valeur $op1)
+	addition	`$op1 + $op2`	-> 7
-	soustraction	`$op1 - $op2`	-> 3
*	multiplication	`$op1 * $op2`	-> 10
/	division	`$op1 / $op2`	-> 2
%	modulo	`$op1 % $op2`	-> 3
-	négatif	`$op1 = -$op2`	-> -2
=	affectation	`$op1 = $op2`	-> 2

3-2 Les opérateurs de comparaison

Vous retrouvez ces opérateurs lors de tests (plus petit, plus grand égal ...).

Opérateur	Signification
==	égalité
<	strictement inférieur
>	strictement supérieur
<=	inférieur ou égal
>=	supérieur ou égal
!= ou <>	différent
===	les opérandes sont égales et de même type

Ne confondez pas le signe d'affectation constitué d'un seul = avec le signe d'égalité constitué de deux ==.

3-3 Les opérateurs logiques

Ces opérateurs permettent de combiner des tests et d'obtenir VRAI ou FAUX en fonctions des valeurs des opérandes. Les opérateurs sont et, ou, ou exclusif et non.

	ET	OU inclusif	OU exclusif
syntaxe	&&	||	xor

Ces opérateurs logiques sont principalement utilisés dans les tests et peuvent bien sûr être

combinés entre eux, tout en prenant garde aux règles de priorités (le ET équivaut à une multiplication et est donc prioritaire sur le OU <=""> addition) Exemple

```
if ($couleur=="jaune") && ($type!="legume") || ($type=="fruit"))
  $objet="banane";
else
  $objet="ce n'est pas une banane";
```

3-4 Les opérateurs « caractères »

La concaténation

Vous pouvez « raccrocher » différents éléments de type caractère et les stocker dans une autre. L'opérateur est :

```
$texte = "cochons";
$variable = "3 petits $texte";
```

Ou

```
$variable = "3 petits".$texte;
```

La comparaison

Vous retrouvez les mêmes opérateurs que pour les numériques.

Utilisations de raccourcis et de combinaisons

Vous pouvez utiliser des raccourcis dans les affectations numériques ou caractères.

Pour les exemples j'utilise les variables initialisées de la manière suivante :

```
$op1=5;
```

Opérateur	Signification	Exemple	Equivalent	Résultat
+=	addition	`$op1 += 7;`	`$opt1 = $opt1 + 7;`	13
-=	soustraction	`$op1 -= 7;`	`$opt1 = $opt1 - 7;`	-2
*=	multiplication	`$op1 *= 7;`	`$opt1 = $opt1 * 7;`	49
/=	division	`$op1 /= 7;`	`$opt1 = $opt1 / 7;`	0
%=	modulo	`$op1 %= 7;`	`$opt1 = $opt1 % 7;`	5

.=	concaténation	`$op1 .= 7;`	`$opt1 = $opt1."7";`	57

Chapitre N°4 : Conditions & Boucles

Les expressions conditionnelles et les boucles

Introduction

Nous avons vu les différents opérateurs utilisables en php. Nous allons maintenant voir les expressions et boucles dans lesquelles se retrouvent généralement ces opérateurs.

4-1 Les expressions conditionnelles

Ces tests permettent de tester la validité des expressions et, en fonction du résultat vrai ou faux, d'effectuer les opérations adéquates.
Ces tests sont des éléments très importants dans la réussite d'un programme. Si vous vous trompez les effets peuvent être catastrophiques. Un OU mis à la place d'un ET dans une expression de test et patatras ...

if

Ce test permet de réaliser une action en fonction de l'état vrai ou faux de l'expression.

```
if(expression)                                          {
  actions1 réalisées si le résultat de l'expression est vrai
}
else                                                    {
  actions2 réalisées si le résultat de l'expression est faux
}
```

Exemple

```
<HTML>
<?
if("$code"                  ==                  "987654")
{
print                                                   "
<META                                  HTTP-EQUIV=REFRESH
CONTENT=\"6;URL=http://www.monsite.com/entreevalide.php\">
<BODY            BGCOLOR=#FFFFFF>              <CENTER>
<IMG                                   SRC=IMAGES/ok.gif>
<BR>
```

```
<FONT                                                SIZE=5>
Bonjour votre mot de passe est valide<BR>
vous allez bientôt pouvroir accéder à notre<BR>
zone                                                privée<BR>
</FONT>
";
}
else
{
print                                                        "
<META                                HTTP-EQUIV=REFRESH
CONTENT=\"6;URL=http://www.monsite.com/erreur.php\">
<BODY          BGCOLOR=#FFFFFF>          <CENTER>
<IMG                        SRC=IMAGES/erreur.gif>
<BR>
<FONT                                                SIZE=5>
Bonjour votre mot de passe est erroné<BR>
veuillez entrez un mot de passe valide<BR>
</FONT>
";
}
?>
<BODY>
</HTML>
```

En fonction du choix de l'utilisateur vous obtiendrez donc deux pages différentes qui redirigeront au bout de 6 secondes vers la page indiquée dans la ligne META ...

Le test *if* propose deux actions en fonction de la valeur de l'expression vraie ou fausse. Vous pouvez imbriquer des *if* permettant ainsi de réaliser des actions en fonction de plus de deux critères. Le problème est que la lisibilité devient vite impossible.

```
if(){}else{if()else{...}}
```

Vous pouvez alors utilisez une alternative grâce au mot clé *elsif*.

```
if                                          (expression)
{
  action1 si la valeur d'expression est vraie
}
elseif                                    (expression2)
{
  action2 si la valeur de expression est fausse et si la valeur
d'expression2                  est                  vraie
}
elseif                                    (expression3)
{
  action3 si la valeur de expression est fausse, la valeur
d'expression2 est fausse et si la valeur d'expression3 est
vraie
}
......
else
{
```

```
  actionn si toutes les valeurs des expressions précédentes
sont                                                fausses
}
```

switch

Cette instruction permet de proposer différentes actions en fonction du résultat d'une expression. Vous me direz le *elseif* fait la même chose. En effet dans le cas où vous évaluez toujours le résultat de la même variable (voir exemple précédent), l'expression est toujours la même c'est vrai. Par contre si vous désiriez évaluer différentes expressions, le switch ne peut convenir.

```
switch                    (expression)                    {
  case            valeur1            :            actions1;
  break;
  case            valeur2            :            actions2;
  break;
  ...
  case            valeurn            ;            actionsn;
  break;
  default         :       actions       par       défaut;
  break;
}
```

Vous voyez que le test ne se fait que sur une expression. En fonction de la valeur de cette expression le programme exécute une des actions. Le mot clé break permet une fois l'action réalisée de sortir du switch. Si vous ne mettez pas le break, les actions suivantes seront éxécutées.

Exemple

```
<HTML>
<?
switch($type)
{
case                    "mathematiques"                    :
  print   "<meta   http-equiv=refresh   content=\"6;
URL=qcmmath.html\">";
break;
case        "histoire       de       l       art"       :
  print   "<meta   http-equiv=refresh   content=\"6;
URL=artqcm.html\">";
break;
case                          "droit"                          :
  print   "<meta   http-equiv=refresh   content=\"6;
URL=droitqcm.html\">";
break;
case                        "ecologie"                         :
  print   "<meta   http-equiv=refresh   content=\"6;
URL=ecoloqcm.html\">";
break;
case             "culture             generale"             :
  print   "<meta   http-equiv=refresh   content=\"6;
URL=cultureqcm.html\">";
break;
default                                                        :
```

```
  print "Si vous ne choisissez rien je fais comment ?";
exit;
}
print "<BODY BGCOLOR=#FFFFFF> <CENTER>
Vous avez choisi le domaine <FONT COLOR=green>$type</FONT>";
?>
<BODY>
</HTML>
```

4-2 Les boucles

Les boucles permettent de répéter les mêmes actions selon la valeur de l'expression indiquée dans le test.
Vous pouvez par exemple, lire le contenu d'un fichier, effectuer des recherches d'occurrence, des opérations mathématiques ...

while

Cette boucle appelée en algorithmique « tant que » permet d'effectuer les mêmes opérations (bloc d'instructions) contenu dans cette boucle tant que la valeur de l'expression est vraie.

Attention le risque de cette boucle est de ne jamais en sortir on appelle cela une boucle infinie. En effet il faut être certain qu'à un moment ou un autre l'expression devienne fausse, si ce n'est pas le cas le bloc d'instructions sera exécuté indéfiniment. Imaginez que ce bloc écrive des informations dans un fichier vous risquez alors la saturation de l'espace disque où se situe ce fichier. Il peut aussi y avoir bien pire comme problème :)

```
while(expression)                                                {
  instructions;
}
```

Exemple

Vous voulez proposer aux internautes les tables de multiplications de leurs choix. Vous leurs demandez quelle table ils veulent (chiffres à multiplier) et juqu'à quel chiffre il désire le voir multiplier.

Page html proposant le formulaire

```
<HTML>
<HEAD>
</HEAD>
<BODY                                          BGCOLOR=white>
<font size=4><U>Bienvenue sur le site de Mathématiques
Plus</U></font>
</CENTER>
<BR>
<FORM ACTION=while.php METHOD="POST">
<LI><FONT COLOR=green SIZE=3>
Choisissez le nombre dont<BR>
vous désirez la table de multiplication
</FONT>
<UL>
```

```
<LI>Donnez le chiffre <INPUT TYPE=TEXT NAME="nombre" SIZE=3>
</UL>
<LI><FONT COLOR=green SIZE=3>Donnez la borne de la table.</FONT>
<BR> (Le nombre doit etre supèrieur à 0)
<UL>
<LI>Donnez la borne <INPUT TYPE=TEXT NAME="borne" SIZE=3>
</UL>
</OL>
<INPUT TYPE=SUBMIT VALUE="Soumettre">
<INPUT TYPE=RESET VALUE="Reset">
</FORM>
</BODY>
</HTML>
```

Programme php traitant le formulaire précédent

```
<html>
<?
print "<font size=3>Vous avez demandez la table de <BR>multiplication
de
<font color=green>$nombre</font> jusqu'à la borne <font color=green>
$borne</font></font><BR><BR>";
print "<table bgcolor=yellow border=1>";
$cpt=0;
while($cpt <="$borne) {
$resultat=$nombre*$cpt;
print "<tr>
<td align=center>$nombre * $cpt</td>
<td align=center>=</td>
<td align=center bgcolor=pink>$resultat</td>
<tr>";
$cpt++;
}
?>
</table>
</body>
</html>
```

Ce programme pour être correct nécessiterait quelques tests de validité des informations entrées par l'utilisateur (borne, chiffre et non caractères ...). Si vous entrez une borne négative aucun tableau ne sera affiché, on n'entre pas dans la boucle.

Dans le programme php l'expression devient fausse dès que le compteur cpt vaut borne + 1. C'est à ce moment que l'on sort de la boucle. Il a donc fallu avant le début de la boucle initialiser le compteur à 0 et surtout incrémenter ce compteur à l'intérieur de la boucle pour le faire évoluer en lui ajoutant 1 à chaque passage. Si vous oubliez cette instruction dans le bloc du while c'est la boucle infinie ... Le reste n'est que de la mise en forme html.

do ... while

Cette boucle ressemble à la boucle précédente while. En algorithmique elle se nomme la boucle « jusqu'à ». A la différence du while, vous exécutez au moins une fois le bloc d'instruction quelque soit la valeur de l'expression. Cette expression n'est évaluée qu'une fois le premier passage réalisé. Le test s'effectue à la fin de la boucle et non au début.

Une fois dans la boucle, les instructions se feront jusqu'à ce que l'expression devienne fausse.

```
do                                                                    {
instructions;
}
while(expresion);
```

Exemple

Nous reprenons le même formulaire que précédemment en modifiant seulement la boucle de traitement dans le programme php.

```
<html>
<?
print "<font size=3>Vous avez demandez la table de <BR>multiplication
de
<font color=green>$nombre</font> jusqu'à la borne <font color=green>
$borne</font></font><BR><BR>";
print     "<table     bgcolor=yellow     border=1>";
$cpt=0;
do
{
$resultat=$nombre*$cpt;
print                                                          "<tr>
<td       align=center>$nombre       *       $cpt</td>
<td                                  align=center>=</td>
<td    align=center    bgcolor=pink>$resultat</td>
<tr>";
$cpt++;
}while($cpt                <="$borne);                ?>
</table>
</body></html>
```

Vous obtiendrez le même résultat qu'avec le while à la différence prêt que si vous donnez une borne négative vous obtiendrez un tableau comportant une première ligne correspondant à la première valeur du compteur soit la multiplication par 0.

for

Cette boucle permet d'effectuer les instructions du bloc lorsque l'on connaît la valeur de la borne. A la différence du while vous n'avez pas à vous soucier de l'arrêt de la boucle, le compteur s'incrémente de lui-même.
Vous utilisez cette boucle lorsque vous connaissez précisément le nombre de fois où vous voulez passer dans le bloc d'instructions. Vous pouvez éventuellement sortir de la boucle en utilisant une commande break comme dans le cas du switch.
Vous pouvez aussi sauter une itération (un passage dans le bloc d'instruction) grâce à la commande continue. Attention dans ces deux cas la lisibilité de votre programme sera plus difficile.

```
for (valeur de départ;valeur de fin de boucle;incrément du compteur)
{
  instructions;
}
```

- *valeur de départ* est la première valeur que prend votre compteur
- *valeur de fin de boucle* est la dernière valeur que prend le compteur (valeur d'arrêt)
- *incrément du compteur* est la valeur que vous ajoutez à chaque passage au compteur

Reprenons le formulaire précédent et modifions le programme php.

```
<html>
<?
print "<font size=3>Vous avez demandez la table de <BR>multiplication
de
<font color=green>$nombre</font> jusqu'à la borne <font color=green>
$borne</font></font><BR><BR>";
print "<table bgcolor=yellow border=1>";
for($cpt=0;$cpt<="$borne;$cpt++) {
$resultat=$nombre*$cpt;
print "<tr>
<td align=center>$nombre * $cpt</td>
<td align=center>=</td>
<td align=center bgcolor=pink>$resultat</td>
<tr>";
}
?>
</table>
</body></html>
```

Vous obtiendrez la même page qu'avec le while. L'initialisation du compteur et son incrémentation sont inclus dans les instructions de la boucle for et non plus dans le bloc de la boucle.

foreach

Une dernière boucle particulière est foreach qui permet de parcourir le contenu d'une liste ou d'un tableau associatif et d'en extraire les couples clé-valeur (voir la partie définition de variables).

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

Ou

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

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
$tab = array( "fred"=>"eric", "dom"=>"tom");
print "Le tableau associatif tab contient";
print " <FONT COLOR=green>".count($tab)."</FONT> élements<BR>";
foreach ($tab as $cle=>$valeur)
{
print "\$tab[<FONT COLOR=green>$cle</FONT>] vaut";
print " <FONT COLOR=green>$tab[$cle]</FONT><BR>";
}
?>
</body></html>
```

Chapitre N°5 : Les Fichiers

Les manipulations de fichiers

Introduction

Les fichiers sont des entités dont on ne peut se passer. En effet, conserver des informations dans le but de les restituer, faire des statistiques, des fichiers d'accès, ... sont des éléments importants que vous utilisez constamment.

5-1 Fonctions de tests

Avant de faire quoique ce soit sur un fichier il faut souvent effectuer des tests pour connaître l'état de celui-ci (existe t'il, est t'il lisible ...). Voici quelques fonctions qui vous simplifieront la vie.

- **file_exists**
  Cette fonction permet de déterminer si un fichier existe et est accessible. Le fichier sous unix par exemple peut exister mais ne pas être accessible par l'utilisateur qui est propriétaire du démon httpd. Elle renvoie la valeur true si le fichier est présent.

```
$fichier="donnees.dat";
if(file_exists($fichier))                    {
  print   "   Le   fichier   $fichier   existe";
}
else
  print "Le fichier $fichier n'existe pas sur la machine
$SERVER_NAME";
```

- **is_file et is_dir**
  Vous permet de définir si l'élément sur lequel on travaille est un fichier. La fonction *is_dir* permet de tester si c'est un répertoire. Ces fonctions renvoient true si c'est un fichier ou si c'est un répertoire.

```
if(is_file("fichier.data"))
  print    "l'élément    est    un    fichier";
else
  print "L'élément n'est pas un fichier";
```

- **is_readable, is_writable et is_executable**
  Cette fonction permet de tester les droits sur le fichier indiqué en paramètre. Attention les droits sont ceux concernant l'utilisateur propriétaire du démon httpd. *is_readable* permet de déterminer si le fichier possède les droits de lecture. *is_writable* permet de déterminer si le fichier possède les droits d'écriture. *is_executable* permet de déterminer si le fichier possède les droits d'exécution.

Exemple

```
<HTML><BODY                    BGCOLOR=#FFFFFF>
<B><U>Test   sur   les   fichiers</U><Br><Br>
<?
#La variable $SCRIPT_FILENAME renvoie le nom du programme en
cours
$variable          =          $SCRIPT_FILENAME;
if(is_file($variable))
{
print   "<BR>Etat   du   fichier   testé<BR>";
# Je lance une commande système qui récupère l'état du fichier
system("ls  -l  $SCRIPT_FILENAME|awk  '{print
$1\"               \"$5\"               \"$9}'");
print "<BR><BR>Test sur le fichier avec les fonctions<BR>";
print "<FONT COLOR=green>$variable</FONT> ";
print     "     est     un     fichier<BR>     ";

if(is_readable($variable))
{print "<FONT COLOR=green>lisible</FONT> ";}

if(is_writable($variable))
{print "<FONT COLOR=green>modifiable</FONT> ";}
if(is_executable($variable))<br> {print "<FONT
COLOR=green>exécutable</FONT>              ";}
```

```
}
else
{
print "<FONT COLOR=green>$variable</FONT> ";
print " n'est pas un fichier ou n'est pas accessible";
}
?>
</FONT></BODY></HTML>
```

- **filesize**
  Vous obtenez avec cette fonction la taille en octets du fichier.

```
$taille=filesize($fichier);
```

- **fileatime, filemtime et filectime**
  Ces fonctions permettent respectivement de déterminer depuis quand le fichier a été accédé, depuis quand le fichier a été modifié et depuis quand le fichier a vu ses attributs modifiés (droits ...).
  La valeur renvoyée correspond au nombre de secondes écoulées depuis le 1er janvier 1970 (c'est le format epoch UNIX).
  Il est donc nécessaire d'utiliser ensuite la fonction *date* pour la remise en forme de cette valeur.

```
$duree=fileatime($fichier);
$duree=filemtime($fichier);
$duree=filectime($fichier);
```

- **touch**
  Pour les amateurs d'unix, cette commande n'est pas inconnue. Elle permet de modifier les attributs de temps sur un fichier en positionnant la date, l'heure ... au moment ou la commande est lancée.
  Si le fichier n'existe pas il permet de le créer à vide.

```
touch($fichier);
```

- **unlink**
  Cette fonction permet de détruire un fichier.
  Vous pouvez aussi utiliser la fonction *system* qui permet de lancer n'importe quelle commande

```
(system("/bin/rm $fichier"))
```

  syntaxe de la fonction:

```
unlink($fichier);
```

- **copy**
  Cette fonction vous permet d'effectuer une copie d'un fichier d'un endroit à un autre.

```
copy($source_fichier,$destination_fichier);
```

5-2 Ouverture et fermeture des fichiers

Maintenant que tous les tests sont effectués, nous allons pouvoir manipuler les fichiers.

- **fopen**
  Nous commencerons par la principale, celle qui permet d'ouvrir un descripteur de fichier pointant sur le premier élément du fichier sur lequel nous voulons travailler. *fopen* renvoie la valeur false si l'ouverture n'a pu avoir lieu, sinon elle renvoie le pointeur (descripteur) référençant le fichier ouvert. Les fichiers peuvent être ouverts en lecture, ajout ou écriture. Pour cela il suffit de positionner le bon attribut. Les attributs sont respectivement *r*,*a* et *w*. Vous pouvez ouvrir un fichier local mais aussi n'importe quel fichier accessible sur le web (via http ou ftp) pour récupérer son contenu (aspirateur de site).

```
$pointeur=fopen($fichier,'attribut');
```

Ou avec un test

```
if($pointeur=fopen($fichier,'attribut'))
{print "Fichier ouvert correctement";}
else
{print "Problème lors de l'ouverture du fichier";}
```

Exemple

```
<HTML> <BODY bgcolor=#FFFFFF>
<B> <U>Travail sur les fichiers</U> <Br>
<br>
<?
$fichier="is_file.php";
if($pointeur=fopen($fichier,'r'))
{
print "Le fichier <font color=green>$fichier</font>
existe<br>";
print "Voici son contenu interprété<br>";
while(!feof($pointeur))
{
#Je lis le contenu HTML du fichier
$ligne=fgets($pointeur,512);
#Je transforme les < et les> pour éviter leurs interprétation
#en HTML sinon nous ne voyons pas le code
$lignemodif1=str_replace("<","&lt;",$ligne);
$lignemodif2=str_replace("<","&lt;",$lignemodif1);
print "<font color=red>$lignemodif2</font><br>";
}
fclose($pointeur);
}
else
{print "Probleme d'ouverture du fichier <font
color=red>$fichier</font>";}
?>
</font> </BODY> </HTML>
```

- **fclose**
  Lorsque vous ouvrez un fichier, une fois son utilisation terminée il faut libérer le pointeur afin de libérer le fichier.

```
fclose($pointeur);
```

5-3 Lecture des données

- **fgets**
  Une fois le fichier ouvert, le but généralement n'est pas d'en rester là, mais de charger le contenu ou d'écrire à l'intérieur. Cette fonction nécessite deux éléments pour pouvoir fonctionner : le pointeur récupéré au moment de l'ouverture du fichier et la taille des blocs de données, en octets, à lire. Si la taille des éléments à lire sur une ligne est inférieure à la limite donnée, le retour chariot (/n) signale la fin du bloc.

```
$ligne=fgets($pointeur,$taille);
```

- **fread**
  Cette fonction fonctionne comme *fgets* à la différence près qu'elle lira toujours des blocs de x octets quelque soit la place du retour chariot. Cette fonction a aussi besoin du pointeur et de la taille en argument.

```
$bloc=fread($pointeur,$taille_bloc);
```

- **fgetc**
  Cette fonction de lecture permet de lire octet par octet le fichier. Elle a besoin du descripteur de fichier en argument (pointeur).

```
$car=fgetc($pointeur);
```

5-4 Ecriture des données

Vous savez comment lire les éléments contenus dans un fichier mais il peut être très intéressant de pouvoir stocker dans des fichiers des informations provenant de formulaires. Pour pouvoir écrire dans un fichier il faut auparavant l'avoir ouvert en écriture ou ajout (voir la partie *fopen*).

fputs et fwrite
Ces deux fonctions fonctionnent de la même manière et renvoient true si l'écriture s'est correctement effectuée. Elles nécessitent comme argument le pointeur de fichier et la chaîne à écrire.

```
fputs($pointeur,$chaine);
fwrite($pointeur,$chaine);
```

Exemple

```
<HTML>                <BODY                bgcolor=#FFFFFF>
<B>    <U>Travail    sur    les    fichiers</U>    <Br>
<br>
<?
$fichier="/tmp/ecrire.txt";
print    "1.    Je    détruis    le    fichier    <font
color=green>$fichier</font><Br>";
unlink($fichier);
print  "2.  J'ouvre  le  fichier  en  ecriture<Br>";
if($pointeur=fopen($fichier,'w'))
{
print    "3.    J'ecris    les    informations<BR>";
fwrite($pointeur,"login:password\n");
fwrite($pointeur,"jacquenod:nantes44\n");
fputs($pointeur,"alexandre:paris75\n");
}
fclose($pointeur);
print  "4.  Je  lis  le  contenu  du  fichier<BR>";
if($pointeur=fopen($fichier,'r'))
{
print "Le fichier <font color=green>$fichier</font> existe<br>";
print        "<HR>Voici        son        contenu<BR>";
$i=0;
while($ligne=fgets($pointeur,512))
{
$i++;
print "ligne$i : <font color=red>$ligne</font><BR> ";
}
fclose($pointeur);
}
else
{print "Probleme d'ouverture du fichier <font
color=green>$fichier</font>";}
?>
</font> </BODY> </HTML>
```

5-5 Autres fonctions utiles

- **feof**
  Cette fonction permet de tester la fin du fichier que vous lisez. Si elle est atteinte, la fonction renvoie la valeur VRAIE. Elle a besoin du descripteur de fichier en argument (pointeur).
  Vous retrouverez *eof* notamment dans les boucles de lecture du contenu d'un fichier.

```
feof($pointeur);
```

- **fseek**
  Cette fonction permet de se positionner dans un fichier. Vous devez indiquer le nombre d'octets que vous parcourez avant de vous positionner. Vous devez comme toujours indiquer le pointeur de fichier.

```
fseek($pointeur,nboctets);
```

- **flock**
  Cette fonction permet de verrouiller un fichier lors de son utilisation par plusieurs programmes. Attention ce verrou n'est valide qu'en PHP4 et seulement dans les programmes php l'utilisant. Si vous combinez un programme php utilisant *flock* avec

un programme perl rien ne sera verrouillé. Vous avez plusieurs niveaux de verrouillage, la lecture simple (1), la lecture et l'écriture interdites (2) et le déverrouillage (3). Pour cela un argument numérique est utilisé en plus du pointeur sur le fichier. Lors du verrouillage seul le programme qui a réalisé le verrouillage peut travailler sur le fichier.

```
flock(pointeur,chiffre);
```

5-6 Importation de fichiers php

Comme dans tous les langages ou presque vous pouvez réutiliser des éléments déjà crées dans un autre programme php sans être obligé de tout réécrire. Pour cela vous devez utiliser le mot clé *include* qui demande comme argument le chemin et le nom du fichier dont les instructions sont à inclure. Ce fichier peut contenir du code HTML brut et ce dernier sera interprété tel quel. Mais vous pouvez inclure un fichier contenant aussi du code PHP pour cela il vous suffit dans ce fichier de créer des instruction PHP incluse entre les bornes du langage PHP (<? ? >).

```
include(fichier);
```

Chapitre N°6 : PHP et MySQL

Interfaçage de MySQL via le langage PHP

Intérêt

Il est très intéressant de pouvoir gérer son site web à travers les bases de données. En effet rien de plus embêtant que de devoir replonger dans le code source pour reprendre les pages à modifier même avec des logiciels WYSIWYG.

Ce serait si bien si l'ajout d'un nouvel article, d'une image ... dans mon site n'obligeait pas à reprendre la page qui propose des liens vers ces domaines. Avec la base de données c'est possible. Vous ajoutez un élément dans une table et c'est tout. Via php, la page proposée à l'utilisateur est construite dynamiquement en effectuant une requête sur la base de données. Les commandes utilisées en php pour réaliser ce que nous faisions manuellement commencent généralement par le mot *MySQL*.

Attention ! Pour suivre ce cours, il est indispensable d'avoir de bonnes notions sur les SGBDR et le langage de requête SQL. De plus, votre hébergeur doit gérer les bases de données MYSQL via le plus souvent PHPMYADMIN.

6-1 Connexion et déconnexion à la base de données

- **Connexion à MySQL**
  Avant de pouvoir effectuer des requêtes sur la base de données, il faut s'y connecter via un utilisateur autorisé.
  On utilise la commande: *mysql_connect*

```
$connexion=mysql_connect
    (nom_machine_locale,utilisateur,mot_de_passe);
```

- **Ouverture de la base**
  Une fois la connexion à MySQL réalisée, il va falloir indiquer quelle est la base de données sur laquelle nous désirons travailler.
  Pour cela utilisez la commande *mysql_select_db*.

```
mysql_select_db(nom_de_la_base,connexion);
```

  Note : le pointeur de connexion serait ici *$connexion* déclaré lors l'ouverture de la connexion.
  On peut outrepasser cette fonction, dans ce cas, il faudra utiliser *mysql_db_query($db,$requete)*

```
$resultat=mysql_db_query($bdd,$requete) or die ('Erreur '.
$requete.' '.mysql_error());
```

  Notez que la fonction *die* est vue dans la gestion des erreurs plus bas sur cette page.

- **Déconnexion**
  Lorsque vous avez terminé d'utiliser la base de données, vous devez, libérer la ressource au moyen de la commande *mysql_close*.

```
mysql_close($connexion);
```

  Si vous omettez cette commande, la fin du script fera office de fermeture de connexion. Si vous omettez le pointeur, ce sera la dernière connexion ouverte qui sera lors fermée.

6-2 État de MySQL

Avant de travailler sur une base contenue dans MySQL vous pouvez vouloir connaître ou faire connaître le contenu de MySQL, permettant ainsi à l'utilisateur de choisir la base sur laquelle il veut travailler (effectuer des requêtes, mise à jour ...). Vous avez à votre disposition la commande *mysql_list_dbs*. Cette commande est identique à *SHOW DATABASES* sous MySql. La commande renvoie un pointeur sur le résultat. Vous n'obtenez pas directement les informations, il faut ensuite utiliser des fonctions de récupération des résultats (*mysql_field_**, *mysql_num_*...) souvent dans une boucle.

```
mysql_list_dbs($connexion);
```

exemple

Je désire fournir à un utilisateur, sous forme de listes, la liste des bases contenues dans MySql. Il doit pouvoir, ensuite, en choisir une et l'ouvrir tout cela avec un seul programme php.

```
<HTML><BODY BGCOLOR=#FFFFFF>
<?php
$username="admin";
$password="motdepasse";
$hostname="monsite.com";
echo ("
<center><font size=7><U>PHP-MYSQL en action<br>Connexion à
MySql</u></font>
<BR><BR>
</CENTER>
");
#On se connecte a MySql
if(!($connexion=mysql_connect
    ($hostname,$username,$password)))
{
echo "<U>probleme lors de la connexion à MySql</U>";
exit();
}
else
{
echo ("<B>Connexion a MySql OK <BR>");
# On recupere les bases presentes
$result=mysql_list_dbs($connexion);
# On compte le nombre de lignes de resultat
$nombre_base=mysql_num_rows($result);
echo "Nous avons <font color=green>$nombre_base</font> bases
disponible(s)<BR>";
# On crée le formulaire qui pointera sur ce meme programme
echo "<FORM METHOD=POST>
<SELECT NAME=nombase SIZE=$nombre_base>
";
#On affiche chaque ligne de resultat
while ($ligne = mysql_fetch_row($result))
{ echo "<OPTION>$ligne[0]";}
echo "
</SELECT><BR>
<INPUT TYPE=submit VALUE=soumettre>
<INPUT TYPE=reset VALUE=reset></FORM>";
# Partie pour le traitement du resultat une fois une base
selectionnee
if($nombase != "")
```

```
{
print "Vous avez choisi d'accéder à la base -->$nombase<--<br>";
# On se connecte a la base
if(!mysql_select_db($nombase,$connexion))
{echo ("Erreur de selection de la base -->$nombase<--<br>");exit();}
else
{echo ("<B>Ouverture de la base -->$nombase<-- ok<br>");}
}
}
mysql_close($connexion);
?>
</BODY></HTML>
```

La partie FORM ne contient pas d'option ACTION. Le programme recevant les informations une fois le formulaire soumis sera donc lui-même. Vous pouvez sinon utiliser la variable d'environnement de php contenant le nom de votre programme:

```
<FORM ACTION=$PHP_SELF METHOD=POST>
```

6-3 État d'une base

Nous connaissons maintenant les bases disponibles, mais qu'y a t'il à l'intérieur ? Nous allons donc avant de voir la récupération d'informations contenues dans les tables comment récupérer l'état d'une base (table, structure ...).

- **Liste des tables**
  Nous l'avons déjà un peu évoqué avec la récupération des bases contenues dans MySql. Vous pouvez utiliser la fonction *mysql_list_tables* associée à la fonction *mysql_tablename*.

```
$pointeur_resultat=mysql_list_tables
     ($nom_de_la_table,$pointeur_de_connexion);
mysql_tablename($pointeur_resultat,$indice);
```

exemple

En reprenant l'exemple précédent, je désire maintenant montrer les tables disponibles dans la base sélectionnée. Afin d'avoir plus d'une table à ma disposition je vais travailler sur la base "baseetudiants". Je complète l'exemple précédent une fois la base ouverte.

```
<HTML><BODY BGCOLOR=#FFFFFF>
<?
else
{
echo ("<B>Ouverture de la base -->$nombase<-- ok<br>");
$result_table=mysql_list_tables($nombase,$connexion);
# On compte le nombre de lignes de resultat
$nombre_table=mysql_num_rows($result_table);
echo "Nous avons <font color=green>$nombre_table</font> tables
disponibles<BR>";
# On crée le formulaire qui pointera su ce meme programme
echo "<FORM ACTION=mysql_query_table.php METHOD=POST>
```

```
<SELECT NAME=nomtable SIZE=$nombre_table>
";
for($j=0;$j<$nombre_table;$j++)
{
echo "<OPTION>".mysql_tablename($result_table,$j);
}
echo "
</SELECT><BR>
<INPUT TYPE=submit VALUE=soumettre>
<INPUT TYPE=reset VALUE=reset>
</FORM>";
}
}
}
mysql_close($connexion);
?>
</BODY></HTML>
```

- **Structure d'une table**

Avant d'effectuer des mises à jour, recherches, insertions d'informations dans une table, il est préférable de bien connaître la structure de la table pour éviter des erreurs.
Vous pouvez utiliser la fonction *mysql_query* qui permet de passer une commande SQL.

```
$pointeur_resultat=mysql_query
    ($requete,$pointeur_de_connexion);
```

exemple

```
$requete="DESC $nomtable";
$result_query=mysql_query($requete,$connexion);
```

Ainsi, la commande *mysql_query* permet de passer n'importe quelle commande comme si vous la tapiez en direct dans MySql. Le résultat est un pointeur référençant un tableau en mémoire.

```
$colonne = mysql_num_fields($result_query);
$ligne = mysql_num_rows($result_query);
```

Je récupère, en utilisant le pointeur de résultat, le nombre de colonnes et de lignes contenues dans le tableau. Ceci me permet ensuite de créer des boucles permettant de lire le contenu de ce tableau. Les indices, je le rappelle commencent à 0.
mysql_num_fields renvoie le nombre de colonnes du tableau résultat (les attributs ici).
mysql_num_rows renvoie le nombre de lignes du tableau résultat (les champs ici).

On peut ensuite afficher le Titre des colonnes (nom des attributs) via la fonction *mysql_field_name* et une boucle *for* qui parcourt le tableau en largeur (nombre de colonnes)

```
for($cpt2=0;$cpt2<$colonne;$cpt2++)
{print "<th bgcolor=yellow>".mysql_field_name
    ($result_query,$cpt2)."</th>";}
```

La fonction *mysql_field_name* permet donc de récupérer le nom de la valeur contenu dans le tableau des résultats. Vous pouvez aussi récupérer la longueur (*mysql_field_length*), le type (*mysql_field_type*) ainsi que les options de l'attribut.

Recuperation des resultats d'une ligne

```
$tab_rows=mysql_fetch_row($result_query);
```

Je récupère, sous forme de liste (*$tab_rows*), chaque ligne de résultats du tableau. A chaque passage, le pointeur se positionne sur la ligne suivante d'où la nécessité d'une boucle.

6-4 Requêtes sur les tables

Voyons maintenant comment effectuer des requêtes de sélections, d'ajout, de créations ... Le principe est assez simple nous allons créer notre requête et ensuite la passer à MySql au moyen de la fonction *mysql_query* déjà vue précédemment.

Récupération d'informations

On souhaite maintenant récupérer toutes les informations contenues dans la table. L'affichage se fera comme pour la structure de la table sous forme de tableau. La requête est simple : *select * from table* Ici on sélectionne tous les enregistrements d'une table, en pratique cela se fait via:

```
$requete="select * from $nomtable";
```

en sachant que l'on peut effectuer une requête plus complexe en fonction des besoins comme :

```
$requete="SELECT * FROM $nomtable WHERE rubrique='chariot' AND type
LIKE                                                    '%$type%'";
$resultat=mysql_db_query($db,$requete) or die ('Erreur '.$requete.'
'.mysql_error());
$n=mysql_num_rows($resultat);
```

Dans ce dernier cas, on utilise la clause *WHERE* pour la requête SQL qui permet de cibler notre demande.

Il nous faut ensuite utiliser une boucle *for* pour lire le contenu de chaque enregistrement, champ par champ.

```
for($i=0;$i<$n;$i++)                                                {
$ligne=mysql_fetch_row($resultat);
$id=$ligne[0];
$description=$ligne[1];

echo ("identifiant n°$id = $description<br>");
}
```

Les différents champs sont présents dans un tableau indicé de 0 à (nombre de champs-1).

Ajout d'informations

Comme indiqué précédemment, on se sert d'une autre requête SQL:

```
$requete="INSERT INTO matable (url,Titre,mail,note,date,logo) VALUES
('$url','$Titresite','$mail',5,NOW(),'$logo')";
$resultat1=mysql_db_query($bdd,$requete) or die ('Insertion dans la
base de données du site '.$Titresite.' impossible '.$requete.'
'.mysql_error());
```

On prend ici l'exemple d'un annuaire de liens où chaque nouveau site est ajouté dans une base de données "matable". On reprend les valeurs de variables renseignées via un formulaire par exemple.
Notez la correspondance rigoureuse des champs qui seront renseignés avec les valeurs indiquées après "VALUES".
On utilise également la fonction MYSQL *NOW()* pour entrer la date actuelle.

Suppression d'informations

On peut également supprimer un ou plusieurs enregistrements en fonction de critères.

```
$requete="DELETE FROM mabase WHERE url='$url'";
$resultat=mysql_db_query($bdd,$requete) or die ('Erreur : le site '.
$Titresite.' n\'a pas été supprimé de la base de données : '.
$requete.' '.mysql_error());
```

Dans cet exemple on utilise donc la fonction *DELETE* et un critère de sélection.

Gestion des erreurs

Deux fonctions permettent, suite à une commande qui a échoué, de récupérer un numéro référençant l'erreur (*mysql_errno*) et un message d'erreur (*mysql_error*).

```
echo ("Numero d'erreur : ".mysql_errno()."<BR>");
echo ("Message d'erreur : ").mysql_error();
```

ou plus simplement

```
print mysql_errno();
print mysql_error();
```

Vous pouvez éventuellement indiquer le pointeur de résultat. Si vous ne le faites pas le dernier récupéré sera alors utilisé.
Une méthode trés utilisée est de stopper le script par la fonction *die* en cas d'erreur.

```
$resultat=mysql_db_query($db,$requete) or die ('Erreur '.$requete.'
'.mysql_error());
```

Informations sur la base

Avec la version php 4.0.5 vous avez à votre disposition des fonctions qui permettent d'obtenir des renseignements sur le client, le serveur ... Ces fonctions renvoient une chaîne de caractères.

mysql_get_client_info() renvoie le numéro de version du client utilisé par PHP
mysql_get_host_info() renvoie le type de connexion utilisée et le nom du serveur hôte
mysql_get_proto_info() renvoie la version du protocole utilisée pour la connexion
mysql_get_server_info() renvoie la version du serveur.

Chapitre N°7 : Les Bases de données

7-1 Accéder aux bases de données avec PHP

Introduction

L'intérêt majeur de PHP est son Interfaçage avec un grand nombre de bases de données d'une manière relativement simple et efficace.
Nous avons vu en introduction que pratiquement tous les SGBD sont pris en charge, mais PHP s'utilise bien souvent avec MySQL, un SGBD rapide (à moyenne charge) et qui satisfait à la plupart des sites Internet, même si il n'a pas encore toutes les potentialités d'autres...
Mais rassurons nous, MySQL est exploitable même pour des bases de données de plusieurs Giga-octets et son moteur est basé sur la norme ANSI SQL 92.
Nous développerons ici les fonctions PHP qui permettent d'exploiter des bases de données MySQL.
Pour la visualisation de vos tables et bases de données MySQL nous vous conseillons l'emploi de l'excellent phpMyAdmin qui est une interface en ligne très simple d'utilisation et redoutable dans ses possibilités.

Périmètre et limites

Nous ne passerons pas en revue les différentes commandes SQL mais nous privilégierons leurs interactions avec PHP. Une bonne connaissance préalable des SGBD et des types de données que l'on y trouve est donc nécessaire.
De plus, il vous appartient de savoir comment on remplit une base de données manuellement pour son exploitation avec PHP.

Les exemples donnés ici s'appuient sur une version de MySQL inférieure à 4.1 qui dispose de nouvelles fonctionnalités et nécessite quelques aménagements. Mais pour le moment et sur pratiquement tous les serveurs, MySQL est installé avec une version antérieure à 4.1.

Utiliser MySQL avec PHP

L'exploitation de MySQL avec PHP s'effectue en 5 étapes :

1. connexion à MySQL
2. sélection de la base de données
3. requête sur la base de données
4. exploitation des résultats de la requête
5. fermeture de la connexion à MySQL

7-2 Se connecter à une base de données MySQL

Avant d'exploiter les données qui se trouvent sur notre base, il nous faut ouvrir un canal de communication qui ne sera fermé qu'une fois toutes nos opérations effectuées. Notre base de données se trouve sur un serveur désigné avec un nom de domaine, un adresse IP ou un alias. Pour nous y connecter, il nous faut un login et un mot de passe. Tous ces paramètres sont fournis par votre hébergeur. Nous choisirons pour nos exemples les variables suivantes :

```
$serveur = "mysql.netalya.com";
$nom_base = "coursPHP";
$login = "cours";
$motdepasse = "php";
```

Ouvrir une connexion

Précisons que **les informations qui vous sont fournies par votre hébergeur sont confidentielles** et ne doivent en aucun cas être communiquées sous peine de voir vos bases modifiées ou détruites par des personnes malintentionnées.

Pour se connecter, on utilise la fonction mysql_connect comme ci-dessous :

```
$serveur = "mysql.netalya.com";
$nom_base = "coursPHP";
$login = "cours";
$motdepasse = "php";
mysql_connect ($serveur,$login,$pwd) or die ('ERREUR
'.mysql_error());
```

ici, nous effectuons un test pour savoir si la connexion est effective ou pas avec l'alias *die()* : en cas d'échec, le code s'interrompera ici en affichant l'erreur rencontrée par MySQL (stockée dans la fonction *mysql_error()*. On pourrait également faire un test plus classique comme ci-dessous :

```
if (mysql_connect ($serveur,$login,$pwd)) {
  echo 'connexion réussie';
}
else {
  echo 'connexion impossible...'.mysql_error();
}
```

Afin d'éviter de définir vos identifiants de connexion à chaque fois que vous exploiter votre base de données, il est utile de les préciser dans un fichier qui sera appelé à chaque fois via un *include()*. Ainsi, vous pourrez facilement les mettre à jour si besoin et renforcerez la sécurité de votre site web (en protégeant l'accès à ce fichier via un .htaccess sur les serveurs Apache).

Sélectionner une base de données

On utilise la fonction *mysql_select_db()* :

```
$serveur = "mysql.netalya.com";
$nom_base = "coursPHP";
```

```
$login = "cours";
$motdepasse = "php";
// connexion à MySQL
mysql_connect ($serveur,$login,$pwd) or die ('ERREUR '.mysql_error());
// sélection de la base de données
mysql_select_db ($nom_base) or die ('ERREUR '.mysql_error());
```

Une fois que notre connexion est établie et notre base de données sélectionnée, nous pourrons effectuer des requêtes et passer à l'étape 3.

Chapitre N°8 : Requêtes sur les bases de données

Effectuer des requêtes sur la base de données MySQL

8-1 Fonctionnement : cas de la clause SELECT

Une fois que nous sommes connectés à MySQL et à une base de données, nous pouvons effectuer des tâches classiques dans un SGBD : ajouter une table, supprimer un enregistrement, mettre à jour un enregistrement, afficher le résultat d'une requête... Toutes ces actions sont directement effectuées en SQL via une requête appropriée et l'utilisation de la fonction PHP *mysql_query ()*. Il est donc indispensable d'avoir au préalable une bonne connaissance du langage SQL... Admettons que nous soyons connectés à notre base de données comme vu précedemment. Nous souhaitons par exemple afficher tous les enregistrements de la table nommée 'table' pour lesquels le champ intitulé 'champ_id' est égal à 'notre-planete.info' :

```
$requete = "SELECT * FROM table WHERE champ_id = 'notre-planete.info'";
$resultat = mysql_query ($requete);
```

Cet exemple suffit pour comprendre comment PHP s'interface avec MySQL : une requête SQL est passée en argument de la fonction *mysql_query ()*, cette requête est ici déclarée avant dans une variable de type chaîne de caractères. De la même façon, on pourrait effectuer une requête de suppression, d'ajout...

Récupérer un enregistrement avec la clause SELECT

Notons que le résultat de la requête se trouve dans la variable nommée *$res*. Ce résultat peut ensuite être exploité comme en témoigne l'exemple ci-dessous :

```
$ligne = mysql_fetch_assoc($resultat);
echo 'le premier enregistrement a pour id '.$ligne["id"].' et pour type '.$ligne["type"];
```

Ce qui donnera :

```
Le premier enregistrement a pour id notre-planete.info et pour type site Internet
```

Pour récupérer les valeurs des champs d'un enregistrement, on utilise donc la fonction *mysql_fetch_assoc()* qui retourne l'enregistrement en cours sous la forme d'un tableau associatif.

Exploiter plusieurs enregistrements avec la clause SELECT

En effet, si notre requête retourne plusieurs enregistrements, il suffit d'effectuer une boucle *while* qui lira tous les enregistrements jusqu'au dernier pour affichage, exploitation... Nous prendrons ici l'exemple de l'affichage des 4 premières actualités du portail **notre-planete.info** présentes en page d'accueil du site. Voici le code utilisé (en admettant que nous sommes déjà connectés à notre base)

```
$requete = "SELECT * FROM table_actualites ORDER BY champ_date DESC LIMIT 4"; // on limite à 4 le nombre d'enregistrements souhaité
//          envoi          de          la          requête
$resultat = mysql_query($requete) or die ('Erreur '.$requete.' '.mysql_error());

// tant qu'il y a un enregistrement, les instructions dans la boucle s'exécutent
while ($ligne = mysql_fetch_assoc($resultat)) {
  $date = date( "d/m", strtotime($ligne['champ_date'])); // on convertit la date dans un format lisible
  $news .= '<br /><font color="#0000ff"><b>'.$date.'</b></font> <a href="http://www.notre-planete.info/actualites/actualites/actu_'.$ligne['champ_numero'].'.php">'.$ligne['champ_actu'].'</a>'; // mise en place d'un lien sur le titre et ajout dans la variable $news
}

echo $news; // affichage de la variable $news qui s'est enrichie à chaque passage dans la boucle d'une actualité
```

Nous constatons que la fonction *mysql_fetch_assoc()* permet de récupérer les valeurs de l'enregistrement en cours tout en passant au suivant une fois la fonction appelée. C'est à dire qu'il n'est pas utile (comme en ASP par exemple) d'indiquer que l'on souhaite maintenant lire le prochain enregistrement, ils sont lus séquentiellement. Notons qu'il n'est pas possible de "revenir en arrière", au premier enregistrement à lire avec cette méthode. Le résultat sera le suivant :

```
04/02    De l'électricité à partir de la pression atmosphérique
03/02    L'effet de serre à la lumière de l'histoire géologique
02/02    Journée mondiale des zones humides
01/02 Les poissons de la Baltique trop toxiques pour être vendus en Europe
```

Avec ces quelques fonctions, nous sommes donc capables d'interfacer facilement PHP avec MySQL, les limites étant celles de vos connaissances en algo et en MySQL ;-)

8-2 Les principales fonctions MySQL en PHP

Fonction	Description
mysql_close()	ferme la connexion MySQL
mysql_connect()	ouvre une connexion à un serveur MySQL
mysql_error()	renvoie l'erreur MySQL rencontrée
mysql_fetch_array()	retourne une ligne de résultat sous la forme d'un tableau associatif
mysql_fetch_assoc()	lit une ligne de résultat dans un tableau associatif
mysql_fetch_row()	découpe une ligne de résultat en un tableau
mysql_free_result()	libère la mémoire du résultat de la dernière requête
mysql_num_rows()	renvoie le nombre d'enregistrements résultants d'une requête de type SELECT
mysql_query()	envoie une requête SQL
mysql_select_db()	sélectionne une base de données MySQL

Les fonctions ci-dessus ont été sélectionnées pour leur emploi courant dans les requêtes aux bases MySQL. Vous trouverez plus de détails sur leurs fonctionnalités sur le site officiel PHP Hypertext Preprocessor où toutes les fonctions PHP sont expliquées et commentées en français :-)

Détaillons quelque peu la fonction *mysql_num_rows()* qui permet de compter le nombre d'enregistrements retournés par une requête de sélection pour mentionner par exemple le nombre de photos de phénomènes naturels disponibles sur le portail **notre-planete.info**

```
$requete = "SELECT * FROM table_photos WHERE champ_categorie='phenomenes_naturels'";
$resultat = mysql_query($requete) or die ('Erreur '.$requete.' '.mysql_error());
$nombre = mysql_num_rows($resultat);
echo $nombre.' photos de phénomènes naturels disponibles !';
```

Ce qui nous donnera :

```
105 photos de phénomènes naturels disponibles !
```

Récapitulons : se connecter et effectuer une requête

- Prenons l'exemple d'une visualisation de toutes les actualités en environnement de l'année en cours sur le portail **notre-planete.info** Nous choisirons ici de privilégier l'inclusion du fichier contenant les identifiants pour se connecter à la base de données, pour les raisons évoquées précédemment

Contenu du fichier id_connect.inc.php :

```
$serveur = "mysql.netalya.com";
$nom_base = "coursPHP";
$login = "cours";
$motdepasse = "php";
mysql_connect ($serveur,$login,$pwd) or die ('ERREUR
'.mysql_error());
```

Inclusion et requête :

```
echo '<table>'; //les actualités seront mises dans un tableau
// on récupère les identifiants de connexion à la base de données via
l'inclusion du fichier id_connect.inc.php
include ($_SERVER["DOCUMENT_ROOT"]."/inc/id_connect.inc.php");

$annee = 2005; //annee souhaitée
$rubrique = "environnement"; //rubrique affichée

//requête de sélection
$requete = "SELECT * FROM champ_actualites WHERE
categorie='$rubrique' AND YEAR(date)=$annee ORDER BY date DESC";
$resultat = mysql_query($requete) or die ('Erreur '.$requete.'
'.mysql_error());
$n = mysql_num_rows($resultat);
for($i=0;$i<$n;$i++) {
  $ligne = mysql_fetch_row($resultat);
  $date = date("d/m", strtotime($ligne[2]));
  $liste .= '<tr><td';
  if ($i%2==0)
    $liste .= ' bgcolor="#eeeeee"';
  $liste .= '><font color="#00216A"><b>'.$date.'</b></font> - <a
href="actu_'.$ligne[0].'.php">'.$ligne[4].'</a></td></tr>';
}

echo $liste.'</table>';
```

Cet exemple complet introduit de nouveaux éléments :

- on peut directement insérer des variables PHP dans la requête ce qui nous donne plus de souplesse avec des requêtes dynamiques qui peuvent par exemple prendre en compte une préférence utilisateur via un formulaire
- l'utilisation de la fonction *mysql_fetch_row* retourne les valeurs de l'enregistrement en cours sous forme d'un tableau indexé avec des nombres. A ce titre, si dans la structure de notre table le premier champ est le "numéro" de l'actualité, on le récupérera via la variable "$ligne[0]" (les indices commencant à 0)

Enfin, voyons comment s'effectuent d'autres types de requêtes sur une base de données : effacement, mise à jour...

Chapitre N°9 : Autres requêtes sur les bases de données

Autres exemples de requêtes possibles

Nous verrons ici, au travers d'exemples, quelques requêtes courantes sur les bases de données.

Supprimer un enregistrement

Exemple pris pour la suppression d'une photo sur le portail **notre-planete.info**. On suppose que la variable photo a été transmise dans l'URL via un lien du type :

```
<a href="supprime_photo.php?photo=lemming.jpg">supprimer la photo</a>
```

La page "supprime_photo.php" exécute alors le code suivant :

```
//    connexion    à    la    base    de    données
include  ($_SERVER["DOCUMENT_ROOT"]."/inc/id_connect.inc.php");

// recupération de la valeur de la variable photo passée dans l'URL
$photo                =                $_GET["asup"];

//   suppression   de   la   photo   sélectionnée
$requete = "DELETE FROM table_photos WHERE champ_photo='$photo'";
mysql_query($requete) or die ('Erreur '.$requete.' '.mysql_error());

//       fermeture       de       la       connexion
mysql_close();
echo 'La photo '.$photo.' a bien été supprimée';
```

Ce script peut servir par exemple pour la suppression de photos qui ne correspondent plus aux exigences de qualité nécessaires des galeries.

Mettre à jour un enregistrement

Exemple pris pour la mise à jour de son CV présent dans la rubrique emploi du portail **notre-planete.info**.
Un demandeur d'emploi peut, sur sa demande, accéder à une page où son CV (déjà posté), est visible et modifiable dans des zones de texte. Une fois qu'il est satisfait de ces modifications, il valide le formulaire et le code suivant s'exécute :

```
//    connexion    à    la    base    de    données
include  ($_SERVER["DOCUMENT_ROOT"]."/inc/id_connect.inc.php");

// récupération des valeurs renseignées dans le formulaire
$numcv              =              $_POST["numcv"];
$titrecv            =            $_POST["titrecv"];
$typecontrat        =        $_POST["typecontrat"];
$contenucv          =          $_POST["contenucv"];

//       mise       à       jour       du       CV
```

```
$requete = "UPDATE table_cv SET champ_titrecv = '$titrecv',
champ_contrat = '$typecontrat', champ_cv = '$contenucv ' WHERE
champ_numero='$numcv'";
mysql_query($requete) or die ('Erreur '.$requete.' '.mysql_error());
```

Bien sûr, il conviendra de mettre en place au préalable, une zone d'authentification pour s'assure que le demandeur est bien le propriétaire du CV...

Insérer un enregistrement

Exemple pris pour l'insertion d'une nouvelle actualité sur le portail **notre-planete.info**. La mise en ligne d'une actualité suppose que l'administrateur du site ait rempli un formulaire comportant une zone de texte "texte" (pour le corps de l'actualité), un choix de boutons radios "rubrique" (pour la rubrique appropriée à l'actualité, une zone de texte "titre", et une liste déroulante "auteur" (pour le choix de l'auteur de cette actualité).

```
// connexion à la base de données
include ($_SERVER["DOCUMENT_ROOT"]."/inc/id_connect.inc.php");

// récupération des valeurs renseignées dans le formulaire
$texte = nl2br($_POST["texte"]); // utilisation de la fonction nl2br
pour tenir compte des sauts de ligne
$rubrique = $_POST["rubrique"];
$titre = $_POST["titre"];
$auteur = $_POST["auteur"];

// selection de l'auteur de l'actualité, dans la table des auteurs
$requete = "SELECT * FROM table_actualites_auteurs WHERE champ_auteur
= '$auteur'";
$resultat = mysql_query($requete) or die ('Erreur '.$requete.'
'.mysql_error());
$ligne = mysql_fetch_row($resultat);
$source = $ligne[0];

// requête d'insertion de l'actualité
$requete ="INSERT INTO table_actualites
(champ_rubrique,champ_date,champ_texte,champ_titre,champ_auteur)
VALUES ('$rubrique',NOW(),'$texte','$titre','$source')";
$resultat = mysql_query($requete) or die ('Erreur '.$requete.'
'.mysql_error());

// récupération du numéro automatique de la nouvelle actualité
$numero = mysql_insert_id();
echo 'Insertion de l\'actualité n°'.$numero.' dans la base MYSQL
réussie !';
```

Ce dernier exemple introduit quelques nouveautés :

- une première requête de sélection est effectuée sur la table "table_actualites_auteurs" afin de récupérer le numéro (clé primaire) de l'auteur, numéro qui sera ensuite inséré dans le champ de la table actualité ("champ_auteur") prévu à cet effet
- lors de l'insertion de la nouvelle actualité, on utilise la valeur NOW() pour remplir le champ "champ_date" avec la date actuelle (au moment de l'exécution du script)
- On utilise un numéro auto unique (clé primaire) qui s'incrémente de 1 pour chaque nouvel enregistrement inséré afin de numéroter les actualités. Il est possible de

récupérer la valeur de ce numéro automatique allouée par MySQL via la fonction mysql_insert_id().

CONCLUSION

Les quelques éléments que nous avons présentés dans ce cours ne sont qu'une base de départ pour explorer toutes les facettes du PHP. Ce langage qui ne cesse de s'enrichir et de s'ouvrir à de nombreux standards, fournit au webmaster une quantité d'outils pour récupérer des informations, en construire (graphiques et PDF à la volée...) d'une façon relativement simple et efficace. Avec ses nombreux atouts, le PHP est devenu sans aucun doute un langage indispensable à connaître et adopter pour le web dynamique, d'autant plus qu'il est reconnu et exploité par les plus importants sites que connaît Internet.

BIBLIOGRAPHIE

Nous vous recommandons vivement de consulter les sources qui m'ont aidé à mettre en place cette présentation et qui vous permettront d'approfondir le PHP :

- Manuel de référence PHP
- Manuel de référence MySQL
- PHP Hypertext Preprocessor. Site officiel qui centralise toutes les fonctions PHP expliquées et commentées en français :-)
- Liste des fonctions PHP
- PHP 5 avancé, E.Daspet, C. Pierre de Geyer - Eyrolles (06/2004).
- PHP professionnel, J.Castagnetto, H.Rawat, S.Schumann, C.Scollo, D.Veliath - Eyrolles (04/2001).
- Le langage PHP (L'Altruiste)

A vous de jouer !