FORMATION au langage PHP-MYSQL

Tunis, 27 Juin – 1 Juillet 2005

Formateur
KEBIR Mohamed Ines

SOMMAIRE

I-Introduction

1- Qu'est-ce que PHP ?

PHP (officiellement "PHP: Hypertext Preprocessor") est un langage de script qui est principalement utilisé pour être exécuté par un serveur HTTP, mais il peut fonctionner comme n'importe quel langage interprété en utilisant les scripts et son interpréteur sur un ordinateur. On désigne parfois PHP comme une plate-forme plus qu'un simple langage.
Sa syntaxe et sa construction ressemblent à celles des langages C++ et Perl, à la différence que le PHP peut être directement intégré dans du code HTML.

Exemple

```
<html>
<head>
<title>Exemple</title>
</head>
<body>
<?
        echo "Bonjour, je suis un script PHP!";
?>
</body>
</html>
```

L'objet de ce langage est de permettre aux développeurs web d'écrire des pages dynamiques rapidement.

2- Fonctionnement :

PHP n'est pas un langage compilé, c'est un langage interprété par le serveur : le serveur lit le code PHP, le transforme et génère la page HTML. Pour fonctionner, il a donc besoin d'un serveur web. De ce fait une plateforme minimale de base pour l'exécution d'un site web développé en PHP comprend :
-l'interpréteur PHP (serveur PHP)
-un serveur web (Apache, IIS, …)

Lorsqu'un visiteur demande à consulter une page Web, son navigateur envoie une requête à un serveur HTTP. Si la page contient du code PHP, l'interpréteur PHP du serveur le traite et renvoie du code généré (HTML).

De ce fait le code PHP n'est jamais visible sur la page finale consultée par le client. Ainsi en éditant le source de la page on n'y trouvera que du code HTML.

3- Langages concurrents pour la génération des sites dynamiques.

JSP, ASP, PYTHON, Perl, coldfusion, CGI

4- Avantages, limitations.

Le langage PHP possède les même fonctionnalités que les autres langages permettant d'écrire des scripts CGI, comme collecter des données, générer dynamiquement des pages web ou bien envoyer et recevoir des cookies, … .

- La plus grande qualité et le plus important avantage du langage PHP est le support d'un grand nombre de bases de données et la simplicité d'interfaçage avec eux.
- PHP est utilisable sur la majorité des systèmes d'exploitation, comme Linux, de nombreuses variantes Unix (incluant HP-UX, Solaris et OpenBSD), Microsoft Windows, Mac OS X, RISC OS et d'autres encore.
- PHP supporte aussi la plupart des serveurs web actuels : Apache, Microsoft Internet Information Server, Personal Web Server, Netscape et iPlanet servers, Oreilly Website Pro server, Caudium, Xitami, OmniHTTPd et beaucoup d'autres encore.
- La gratuité et la disponibilité du code source (PHP est distribué sous licence GNU GPL)
- La simplicité d'écriture de scripts (apprentissage rapide);

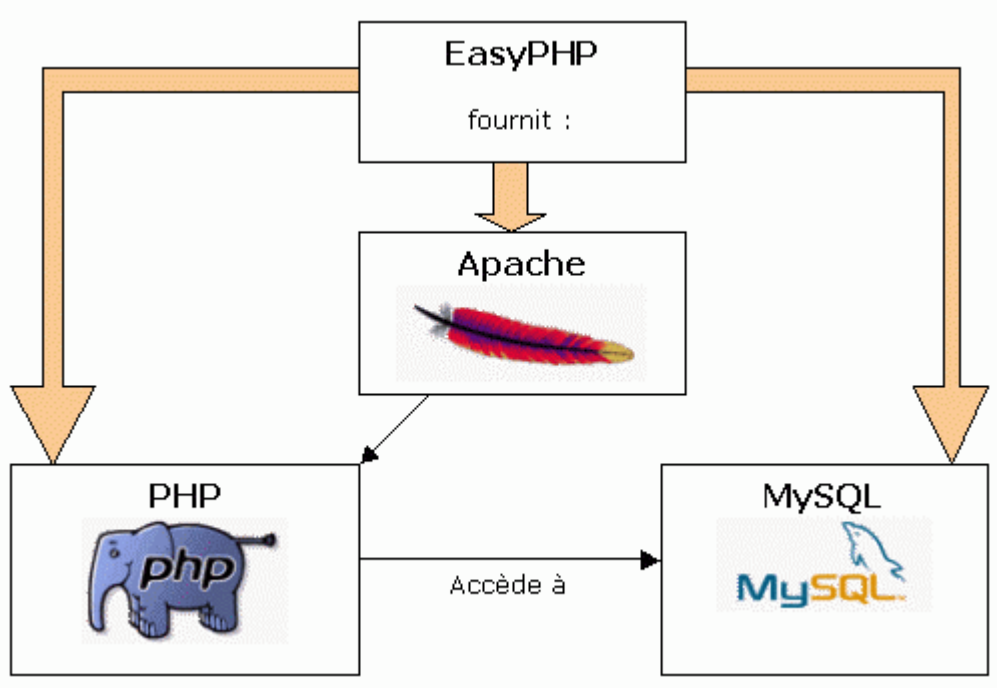
Limitations :

- Langage interprété
- L'orienté objet reste limité
- Pas adéquat en terme de rapidité et de maintenabilité pour des projet de grandes envergures

II-Mise en œuvre et déploiement

1- EasyPHP

EasyPHP est un paquetage contenant à la fois Apache, PHP et MySQL
EasyPHP permet d'installer automatiquement et facilement une plateforme permettant l'exploitation d'un site web en PHP qui éventuellement aurait besoin d'un accès a une base de données. De la sorte on se libère des complications d'une installation manuelle de chacune de composantes séparément.



Procédure d'installation :

- Télécharger EasyPHP sur le site www.easyphp.org
- Double cliquer sur l'exécutable téléchargé
- Sélectionner le répertoire d'installation et suivre la procédure

Procédure de lancement :

Lorsqu'EasyPHP est lancé, les serveurs Apache et MySQL sont automatiquement lancés (il est même possible de le faire automatiquement au démarrage de Windows). Une petite icône s'installe dans la barre des tâches, à côté de l'horloge, permettant un accès rapide aux fonctions proposées par EasyPHP :

- Fichier Log : renvoie aux erreurs générées par Apache et MySQL
- Configuration : donne accès aux différentes configurations d'EasyPHP
- Administration : gestion des répertoires virtuels, des extensions, de PHPMyadmin
- Web local : ouvre la page http://localhost/ répertoire racine du site
- Démarrer/Arrêter : démarre/arrête Apache et MySQL
- Quitter : ferme EasyPHP

Pour que vos pages PHP soient interprétées, il est impératif de placer vos fichiers dans le répertoire www (sus le répertoire d'installation de easyphp). Ce répertoire est configuré par défaut comme répertoire racine de votre serveur web. Le serveur Apache est configuré pour ouvrir automatiquement un fichier index lorsque vous saisissez l'adresse 'http://localhost/' (à condition évidemment que le serveur Apache soit en route). Cette page sert de page d'accueil au web local et permet de vérifier le bon fonctionnement d'EasyPHP.
Il est conseillé de créer un répertoire par projet dans le répertoire www afin d'avoir une vision plus claire des développements.

Il est en outre possible de créer des répertoires virtuels. Ces répertoires virtuels peuvent être créé physiquement n'importe ou sur votre disque dur. Il suffira de leur associer un nom virtuel avec lequel le serveur apache le reconnaîtra.

2- Configuration et paramétrage (php.ini)

Lors de l'installation de PHP, un fichier de configuration nommé « php.ini » sera créé sur votre système. Le fichier php.ini est le fichier de configuration de serveur PHP. Ce fichier respecte la structure des fichiers INI bien connus de nombreuses applications Windows. Il s'agit d'un fichier texte ASCII divisé en plusieurs sections, chacune portant un nom et contenant des variables relatives à la section concernée.
Chaque section ressemble au fragment suivant :

```
[MaSection]
variable="valeur"
autrevariable="autrevaleur"
```

Le nom de la section figure au début entre crochets, suivi d'un certain nombre de paires nom/valeur, chaque paire figurant sur une ligne distincte. Comme pour tout code PHP, les noms des variables font la distinction entre majuscules et minuscules et ne peuvent pas contenir d'espace. Les valeurs peuvent quant à elles être numériques, booléennes ou correspondre à une chaîne.

Les lignes en commentaires commencent par un « ; »

Toute modification du fichier php.ini nécessite un redémarrage du serveur afin que la nouvelle configuration soit prise en compte.

Par défaut le fichier php.ini est configuré de sorte à être fonctionnel. Intervenir sur ce fichier peut être intéressant pour rajouter de nouveaux modules ou pour affiner certains paramètres de sécurité mais doit systématiquement faire l'objet d'un backup de sécurité au préalable.

Quelques exemples de paramétrage :

-Gérer l'affichage des messages d'erreur

Les erreurs PHP se répartissent en quatre catégories: erreurs du parseur, notifications de bugs dans le code (par exemple, des variables non initialisées), avertissements (erreurs non fatales) et erreurs fatales. Normalement, lorsque PHP détecte une erreur de parseur, non fatale ou fatale, il affiche l'erreur et, si elle est fatale, il arrête également de traiter le script à ce stade. Vous pouvez modifier ce comportement à l'aide de la variable error_reporting, qui accepte un champ binaire des codes d'erreurs et n'affiche que les erreurs correspondant à ces codes:
La configuration suivante par exemple permet l'affichage de tout type d'erreur.

```
error_reporting =  E_ALL
```

Pour désactiver l'affichage des erreurs (recommandé en production), désactivez la variable display_errors et inscrivez plutôt les messages dans un journal des erreurs à l'aide de la variable log_errors.

Cette technique est également efficace pour des raisons de sécurité: en désactivant les erreurs, vous masquez des informations spécifiques du système que des utilisateurs peu scrupuleux pourraient exploiter pour essayer de nuire à votre site ou votre application. Définissez donc l'écriture de ces erreurs dans un fichier journal personnalisé ou dans le journal système, en paramétrant la variable error_log sur le nom d'un fichier .

```
display_errors = Off
log_errors = On
error_log = "error.log"
```

- file_uploads= On/Off permet d'autoriser ou non l'envoi de fichiers.

- upload_tmp_dir = répertoire permet de définir le répertoire temporaire permettant d'accueillir le fichier uploadé.

- upload_max_filesize = 2M permet de définir la taille maximale autorisée pour le fichier. Si cette limite est dépassée, le serveur enverra un code d'erreur.

- post_max_size indique la taille maximale des données envoyées par un formulaire. Cette directive prime sur upload_max_filesize, il faut donc s'assurer d'avoir post_max_size supérieure à upload_max_filesize.

- register_globals = Off [sécurité, performance]
Depuis la version 4.2.0 de PHP, la valeur par défaut de register_global est à Off dans le php.ini. Dorénavant une variable envoyée par un formulaire (méthode POST) n'est plus récupérée avec $variable mais avec $_POST["variable"]. Toutes les variables globales sont concernées (POST, GET, cookies, environnement et autres variables serveur : $_GET, $_POST, $_COOKIE, $_SERVER, $_ENV, $_REQUEST, $_SESSION). Ceci peut nécessiter la réécriture partielle de certains scripts.
Rq : il est vivement conseillé d'utiliser cette configuration qui est celle adoptée par défaut depuis PHP 4.2.0 et de coder vos scripts en conséquence. Cependant si vous souhaitez utiliser d'anciens scripts sans avoir à les réécrire, vous avez toujours la possibilité de remettre dans le fichier php.ini register_global à On.

3- Présentation des options les plus courantes.

Le Menu de EASYPHP offre les fonctionnalités suivantes :

- Fichier Log

Ce menu permet de visualiser l'ensemble des fichier logs que Apache et MYSQL génèrent. PHP ne générant pas de fichier log puisqu'il ne fonctionne qu'au travers d'Apache.

- Configuration

Ce menu donne accès aux fichiers de configurations suivants :
-Apache : httpd.conf
-Extension PHP : php.ini
-PHP : php.ini
-MYSQL : mysql.ini
-PHPMyadmin

- Administration

-Ce menu va permettre entre autre de créer des répertoires virtuels pour les sites web que vous allez créer.
-Il permettra en outre d'accéder à PHPMyadmin afin de gérer vos bases de données.

- Web local
- Démarrer/Arrêter
- Quitter

III-Les fonctionnalités du langage

1- Premiers éléments du langage.

Lorsque vous créez un code, il doit être placé entre balises php pour que celui- ci soit interprété, comme ceci:

```
<?
echo 'bonjour';
?>
ou encore
<?PHP
echo 'bonjour';
?>
Ce qui affichera à l'écran ' bonjour '.
```

Une syntaxe PHP se termine TOUJOURS par un point-virgule, si vous l'oubliez vous verrez apparaître une PARSE ERROR lors de l'exécution de votre fichier.

Les commentaires peuvent se présenter sous deux formes :

```
<?
//ceci est
// un commentaire

/* ceci est
un commentaire */
?>
```

2- Intégration de PHP dans une page HTML.

a- Généralités

L'un des avantages du PHP est qu'il s'intègre facilement dans du code HTML classique. Chacun peut à sa guise inclure quelques parties en PHP dans des parties de code HTML.

Remarque importante :
Du moment que des parties de code PHP ont été intégrées dans une page HTML cette dernière doit impérativement être renommée en extension **.php**

Exemple

```
<html>
<body>

<font size="2" face="Arial">Le texte en HTML</font> <br>

<?
echo "<font size=\"2\" face=\"Arial\"> Le texte est en PHP.</font>";
?>

<br>
<font size="2" face="Arial"> < ? echo 'Encore du texte en PHP' ?></font>

</body>
</html>
```

Le résultat obtenu sera :
Le texte est en HTML
Le texte est en PHP
Encore du texte en PHP

b- La fonction include()

La fonction **include()** de PHP permet d'inclure un fichier dans un autre lors de son exécution.
Un élément HTML répétitif qui apparaît dans toutes les pages de votre site (menu par exemple) pourra être isolé dans un fichier PHP. Un appel de ce fichier grâce à la fonction **include()** apparaîtra dans toutes les pages de votre site. Ainsi si le menu doit être par exemple modifié il suffira uniquement de changer le fichier contenant le menu.

Syntaxe

```
<?
Include ("nom_du_fichier_a_inclure") ;
?>
```

Dans l'exemple suivant un fichier menu.php contiendra le code HTML nécessaire à la génération d'un menu en HTML.
Un second fichier page.php inclura ce fichier menu.php.

Exemple fichier menu.php

```
<a href= "menu1.php" > Menu1 </a><br>
<a href="menu2.php" > Menu2 </a><br>
<a href="menu3.php" > Menu3 </a><br>
```

Fichier page.php

```
<html>
<body>
<?
Include ("menu.php") ;  / /inclusion du fichier contenant le menu
?>
</body>
</html>
```

Le résultat obtenu sera :

Menu1
Menu2
Menu3

Remarque : Il ne faut exécuter que le fichier page.php.

Ainsi toutes les pages du site sensées contenir ce menu intègreront grâce à la fonction include() le fichier menu.php.
Il suffira de changer le fichier menu.php afin que toutes les pages qui l'intègrent se trouvent automatiquement changées.

3- Variables, chaînes et concaténation

Une variable est définie sous la forme *$variable_nom.*
L'affectation d'une variable se fait de la manière suivante : $variable_nom = variable_valeur.
L'appel de la valeur affectée à une variable se fait par son nom.

Syntaxe

```
<?
$variable1 = 'Bonjour 1';          //Affectation d'une chaîne avec quote simple

$variable2 = "Bonjour 2";          // Affectation d'une chaîne avec quote double

$variable3 = 5;                    // Affectation d'un entier

$variable4 = 2 + (3 * 5);          // Affectation d'un résultat d'opération

$variable5 = true;                 // Affectation Booléenne

?>
```

L'affichage des variables combinées à des chaînes de caractères peut se faire de plusieurs manières en utilisant les cotes simples (') ou les doubles cotes (").

```
<?
$nom = 'visiteur';
echo "bonjour $nom";
?>
```

Le résultat obtenu sera :
bonjour visiteur

```
<?
$nom = 'visiteur';
echo 'bonjour '.$nom;
?>
```

Le résultat obtenu sera :
bonjour visiteur

Remarque : Il y'a un . entre le ' et la variable $nom. Le point est un opérateur de concaténation pour les chaînes de caractères.

```
<?
$nom = 'visiteur';
echo 'bonjour $nom';   // La variable $nom ne sera pas interprétée
?>
```

Le résultat obtenu sera :
bonjour $nom

Remarques :
Si vous utilisez les ' au lieu des " , la variable n'est pas interprétée comme une variable mais comme une chaîne de caractère.

Remarque sur l'usage des cotes simples ou doubles
Lors de l'usage des ' pour l'affichage d'une chaîne de caractère contenant des apostrophes, vous devez impérativement faire précéder ces apostrophes d'antislash. De la sorte l'apostrophe ne sera pas confondue avec le caractère de fin de la chaîne à afficher. Dans le cas contraire un message d'erreur sera affiché.

```
<?
echo 'vous n|'êtes pas inscrit';
?>
```

Le résultat obtenu sera :
Vous n'êtes pas inscrit

Il en va de même lors de l'usage de '' pour l'affichage d'une chaîne de caractère contenant au préalable des doubles cotes.

```
<?
echo"<a href=|"http://www-etu|">lien.php</a>";
?>
```

Le résultat obtenu sera :
Lien.php

4- Conditions et boucles.

a- Conditions if

La syntaxe de base d'une instruction conditionnelle est la suivante :

```
<?
if($var == 'condition')
{
// 'condition vérifiée';
}
else
{
//'condition non vérifiée';
}
?>
```

Les opérateurs de contrôle sont les suivants :

==	strictement égal
!=	différent
>	plus grand que
<	inférieur à
>=	supérieur à
<=	inférieur à
&&	et
||	ou
AND	et
OR	ou
TRUE	1 ou oui
FALSE	0 ou non

Exemple

```
<?
$montant='100';
if ($montant <1000)
{
echo 'Montant inférieur à 1000';
}
else
{
echo 'Montant supérieur à 1000';
}
?>
```

Le résultat obtenu sera :
Montant inférieur à 1000

b- Conditions elseif

```
<?
If ($var == 'condition1')
{
        // 'condition1 vérifiée';
}
elseif  ($var == 'condition2')
{
        // 'condition2 vérifiée';
}
…
elseif  ($var == 'conditionN)
{
        // 'conditionN vérifiée';
}
…
else
{
        echo 'Aucune condition n'est vérifiée';
}
?>
```

Les structures **elseif** pouvant se répéter autant de fois que des conditions prévues l'exigeront.

Exemple 1

```
<?
$montant='100' ;
if ($montant >=0 && $montant<1000)
{
        echo ' Votre montant '. $montant. ' est compris entre 0 et 1000';
}
elseif ($montant >=1000 && $montant<5000)
{
        echo ' Votre montant '. $montant. '  est compris entre 1000 et 5000';
}
else
{
        echo ' Votre montant '.$montant. ' est supérieur à 5000';
}
?>
```

Le résultat obtenu sera :
Votre montant 100 est compris entre 0 et 1000

Exemple 2

```
<?
$variable = 3;

// serie de tests
if ($variable == 1)
   {
   echo 'La variable a pour valeur 1';
   }
elseif ($variable == 2)
   {
   echo 'La variable a pour valeur 2';
   }
elseif ($variable == 3)
   {
  echo 'La variable a pour valeur 3';
   }
elseif ($variable == 4)
   {
  echo 'La variable a pour valeur 4';
   }
elseif ($variable == 5)
   {
   echo 'La variable a pour valeur 5';
   }
else
   {
   echo 'La variable n'appartient pas à l'intervalle [1,5]';
   }
?>
```

Le résultat obtenu sera :
La variable a pour valeur 3

c- Conditions SWITCH

Le switch est une structure qui permet d'éviter la lourdeur de l'écriture de l'exemple 2. Elle permet en outre une meilleure lisibilité.

Syntaxe

```
<?
switch ($variable)
   {
   case condition1:
      //Traitement de la condition 1
      break;
   case condition2:
      //Traitement de la condition 2
      break;

 ….

   case conditionN:
      //Traitement de la condition N
      break;
   default:
       //Traitement par défaut
   }
?>
```

Exemple

```
<?
$variable = 3;

switch ($variable)
{
   case 1:
      echo 'La variable a pour valeur 1';
       break;
case 2:
      echo 'La variable a pour valeur 2';
       break;
case 3:
      echo 'La variable a pour valeur 3';
       break;
case 4:
      echo 'La variable a pour valeur 4';
       break;
case 5:
      echo 'La variable a pour valeur 5';
       break;
default:
        echo 'La variable n'appartient pas à l'intervalle [1,5]';
   }
?>
```

Le résultat obtenu sera :
La variable a pour valeur 3

d- Itération avec WHILE

Syntaxe

```
<?
 While (condition)
   {
   //Traitements
   }
?>
```

Exemple

```
<?
$nbre_maximum = 6;
$i = 0;                //initialisation de l'indice d'incrémentation
while ($i < $nbre_maximum)   //condition
   {
   echo  $i.' est inférieur à '. $nbre_maximum.'<br>';
   $i++;  // $i++ est équivalent à ($i+1)
   }
echo  $i.' est égal à '. $nbre_maximum;
?>
```

Le résultat obtenu sera :
0 est inférieur à 6
1 est inférieur à 6
2 est inférieur à 6
3 est inférieur à 6
4 est inférieur à 6
5 est inférieur à 6
6 est égal à 6

e- Itération avec FOR

Syntaxe

```
<?
for($i=0; $i != condition ; $i++)
   {
   //Traitements réalisés
   }
?>
```

Exemple

```
<?
$nbre_maximum = 6;

//----------DEBUT BOUCLE------------
for($i=0; $i != $nbre_maximum ; $i++)
   {
   echo  $i.'est inférieur à '. $nbre_maximum.'<br>';
   }
//------------FIN BOUCLE------------

echo  $i.' est égal à '. $nbre_maximum;
?>
```

Le résultat obtenu sera :
0 est inférieur à 6
1 est inférieur à 6
2 est inférieur à 6
3 est inférieur à 6
4 est inférieur à 6
5 est inférieur à 6
6 est égal à 6

5- Fonctions

PHP propose une palette approximative de 2000 fonctions prédéfinies. Toutefois il vous est possible de créer vos propres bibliothèques de fonctions pour vos usages spécifiques, une meilleure lisibilité du code et une réutilisation.

a- Définition des fonctions

Les fonctions peuvent se distinguer en deux sous groupes :

-les fonctions qui effectuent un traitement (affichage par exemple)
-les fonctions qui effectuent un traitement et retournent un résultat

Syntaxe

```
function nom_de_la_fonction ($paramètres)
{
//traitement sur les paramètres effectué
}
```

L'appel de la fonction se fait de la manière suivante :

```
nom_de_la_fonction ($paramètres) ;
```

Exemple

```
<?
function afficher_nom_prenom ($nom,$prenom)
{
echo 'Bonjour '.$nom. ' '.$prenom ;
}

afficher_nom_prenom ('Tarak','Joulak') ;
echo ' <br>' ;

$nom1='Mourad' ;
$prenom1='Zouari' ;
afficher_nom_prenom ($nom1,$prenom1) ;
?>
```

Le résultat obtenu sera :
Bonjour Tarak Joulak
Bonjour Mourad Zouari

Dans le cas suivant un traitement est effectué à l'intérieur de la fonction puis retourné par cette dernière. De ce fait la fonction doit donc être affectée à une variable.

Syntaxe

```
function nom_de_la_fonction ($paramètres)
{
//traitement sur les paramètres effectué
return ($resultat) ;
}
```

L'appel de la fonction se fait de la manière suivante :

```
$variable = nom_de_la_fonction ($paramètres) ;
```

Exemple

```
<?
function additionner ($variable1,$variable2)
{
$total = $variable1 + $variable2;
return ($total) ;
}
$resultat= additionner (1,2) ;
echo  $resultat.' <br>' ;

$var1=6 ;
$var2=7 ;

$resultat= additionner ($var1,$var2) ;
echo  $resultat.' <br>' ;
?>
```

Le résultat obtenu sera :
3
13

b- Librairie de fonctions

Idéalement toutes les fonctions créées devraient être regroupées dans un même fichier créant ainsi une bibliothèque de fonctions. Ce fichier sera appelé à l'intérieur des autres fichiers par le biais de la fonction **include**.

Exemple fichier fonction.inc.php

```
<?
//Ce fichier contiendra l'ensemble des fonctions que vous développerez

function additionner ($variable1,$variable2)
{
$total = $variable1 + $variable2;
return ($total) ;
}

function afficher_nom_prenom ($nom,$prenom)
{
echo 'Bonjour '.$nom. ' '.$prenom ;
}
?>
```

Exemple fichier **page.php**

```
<?
Include ("fonction.inc.php") ;

$resultat= additionner (1,2) ;
echo  $resultat.' <br>' ;

afficher_nom_prenom ('Tarak','Joulak') ;

?>
```

Le résultat obtenu sera :

3
Bonjour Tarak Joulak

6- Tableaux

a- Tableaux numérotés et tableaux associatifs

Il existe deux type de tableaux de variables sous PHP :

-Les tableaux à index numériques (tableaux numérotés) dans lesquels l'accès à la valeur de la variable passe par un index numérique
ex : $tableau[0], $tableau[1], $tableau[2], …

-Les tableaux à index associatifs (ou tableaux associatifs) dans lesquels l'accès à la valeur de la variable passe par un index nominatif
ex : $tableau[nom], $tableau[prénom], $tableau[adresse], …

Syntaxe

```
//Tableau à index numéroté
$tableau = array (valeur0,valeur1,valeur2, …) ;

//Accès à chacune des valeurs
$tableau[0] donnera valeur0
$tableau[1] donnera valeur1
…

//Tableau à index associatif
$tableau = array (variable1 => valeur1, variable2 => valeur2, …) ;

//Accès à chacune des valeurs
$tableau[variable1] donnera valeur1
$tableau[variable2] donnera valeur2
…
```

Exemple

```
<?
//Tableau à index numéroté
$tableau1 = array ('château','maison','bateau') ;
echo "Contenu du tableau 1 :<br>";
echo $tableau1[0]."<br>";
echo $tableau1[1]."<br>";
echo $tableau1[2]."<br>";

//Tableau à index associatif
$tableau2 = array ('prenom' =>'Tarak','nom' =>'Joulak','ville' =>'Tunis') ;
echo "Contenu du tableau 2 :<br>";
echo $tableau2['prenom']."<br>";
echo $tableau2['nom']."<br>";
echo $tableau2['ville']."<br>";
?>
```

Le résultat obtenu sera :

```
Contenu du tableau 1 :
château
maison
bateau
Contenu du tableau 2 :
Tarak
Joulak
Tunis
```

b- Parcours des tableaux

PHP intègre une structure de langage qui permet de parcourir un à un les élément d'un tableau : **foreach().**

Syntaxe

```
$tableau = array (valeur0,valeur1,valeur2, …) ;

foreach ( $tableau as $valeur )
{
//Appeller ici la valeur courante  par $valeur
…
}
```

Exemple

```
<?
//Cas du tableau numéroté
$tableau1 = array ('chateau','maison','bateau') ;
foreach ( $tableau1 as $valeur )
{
  echo $valeur."<br>";
}
?>
```

Le résultat obtenu sera :

château
maison
bateau

Exemple

```
<?
//Cas du tableau associatif
$tableau2 = array ('prenom' =>'Tarak','nom' =>'Joulak','ville' =>'Tunis') ;
foreach ( $tableau2 as $valeur )
{
  echo $valeur."<br>";
}
?>
```

Le résultat obtenu sera :

Tarak
Joulak
Tunis

Dans l'exemple suivant et pour le cas des tableaux associatifs nous pouvons grâce à la boucle **foreach()** aussi bien énumérer le nom des variable que leur valeur.

Exemple

```
<?
$tableau2 = array ('prenom' =>'Tarak','nom' =>'Joulak','ville' =>'Tunis') ;
foreach ( $tableau2 as $variable=>$valeur )
{
  echo $variable." a pour valeur ".$valeur."<br>";
}
?>
```

Le résultat obtenu sera :

prenom a pour valeur Tarak
nom a pour valeur Joulak
ville a pour valeur Tunis

c- Recherche dans un tableau

- array_key_exists() permet de vérifier si dans un tableau associatif une variable associative (clef) existe ou non. Elle retourne donc une valeur booléenne (True ou False).

Syntaxe

```
$resultat= array_key_exists( 'nom_de_la variable', tableau_associatif) ;
```

Exemple

```
<?
$tableau2 = array ('prenom' =>'Tarak','nom' =>'Joulak','ville' =>'Tunis') ;

if (array_key_exists("nom", $tableau2))
{
   echo 'La variable "Nom" se trouve dans le tableau et a pour valeur:
'.$tableau2['nom'];
}
else
{
   echo 'La variable "Nom" ne se trouve pas dans le tableau';
}

?>
```

Le résultat obtenu sera :
La variable "Nom" se trouve dans le tableau et a pour valeur: Joulak

- in_array() permet de déterminer si une valeur existe dans le tableau. Elle retourne donc une valeur booléenne (True ou False).

Syntaxe

```
$resultat= in_array ( nom_de_la_valeur, tableau) ;
```

Exemple

```
<?
$tableau2 = array ('prenom' =>'Tarak','nom' =>'Joulak','ville' =>'Tunis') ;

if (in_array("Tarak", $tableau2))
{
   echo 'La valeur "Tarak" existe bien dans le tableau';
}
else
{
   echo 'La valeur "Tarak" ne se trouve pas dans le tableau';
}

?>
```

Le résultat obtenu sera :
La valeur "Tarak" existe bien dans le tableau

-array_search fonctionne comme in_array : il travaille sur les valeurs d'un tableau.
Si il a trouvé la valeur, array_search renvoit la clé correspondante (c'est-à-dire le numéro si c'est un tableau numéroté, ou le nom de la variable si c'est un tableau associatif). Si il n'a pas trouvé la valeur, array_search renvoit false (comme in_array).

Syntaxe

```
$resultat= array_search ( nom_de_la_valeur, tableau) ;
```

Exemple

```
<?
$tableau2 = array ('prenom' =>'Tarak','nom' =>'Joulak','ville' =>'Tunis') ;

if ($position = array_search("Tunis", $tableau2))
{
        echo '"Tunis" se trouve en position ' . $position . '<br />';
}
else
{
        echo '"Tunis" ne se trouve pas dans le tableau.';
}
?>
```

Le résultat obtenu sera :
"Tunis" se trouve en position ville

7- Variables serveur

PHP propose l'accès à toute une série de variables comme les en-têtes, dossiers et chemins des scripts, sans que vous ayez à les créer, on les appelle les variables serveur. Ces variables sont créées par le serveur web.

Le tableau $_SERVER permet d'accéder à ces variables. Si la directive register_globals de PHP.INI est active, alors ces variables seront aussi rendues directement accessible dans le contexte d'exécution global c'est à dire séparément du tableau $_SERVER.

Description de certaines variables :

$_SERVER['SERVER_NAME'] Le nom du serveur hôte qui exécute le script
$_SERVER['PHP_SELF'] Le nom du fichier du script en cours d'exécution, par rapport à la racine web
$_SERVER['DOCUMENT_ROOT'] Racine du serveur
$_SERVER['REMOTE_ADDR'] Adresse IP du client
$_SERVER['QUERY_STRING'] Liste des paramètres passés au script
$_SERVER['REQUEST_METHOD'] Méthode d'appel du script

Exemple

```
<?
echo 'PHP_SELF: '.$_SERVER['PHP_SELF'];
echo "<br>";
echo 'SERVER_NAME: '.$_SERVER['SERVER_NAME'];
echo "<br>";
echo 'DOCUMENT_ROOT: '.$_SERVER['DOCUMENT_ROOT'];
echo "<br>";
echo 'REMOTE_ADDR: '.$_SERVER['REMOTE_ADDR'];
echo "<br>";
echo 'QUERY_STRING: '.$_SERVER['QUERY_STRING'];
echo "<br>";
echo 'REQUEST_METHOD: '.$_SERVER['REQUEST_METHOD'];
echo "<br>";
?>
```

Le résultat obtenu sera :

```
PHP_SELF: /page.php
SERVER_NAME: localhost
DOCUMENT_ROOT: c:/program files/easyphp1-8/www
REMOTE_ADDR: 127.0.0.1
QUERY_STRING:
REQUEST_METHOD: GET
```

La fonction PHP permettant d'éditer l'intégralité des variables prédéfinies disponibles est phpinfo()

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

IV- Accès fichiers

1- Ouverture / fermeture de fichier

Avant toute opération de lecture ou écriture sur un fichier il y a nécessité de l'ouvrir.
Et à la fin de tout traitement d'un fichier il y a nécessité de le fermer.

Syntaxe

```
<?
// Ouverture de fichier
$monfichier = fopen("nom_du_fichier", "r+");

// Traitements sur le fichier
// .....

// Fermeture du fichier
fclose($monfichier);
?>
```

Fopen() prend en entrée :
- -le nom du fichier (ou meme une url)
- -le mode d'ouverture du fichier

Il retourne un handle.

Fclose() prend en entrée le handle envoyé par le fopen.

Les modes d'ouverture d'un fichier sont les suivants :
'r' Ouvre en lecture seule, et place le pointeur de fichier au début du fichier.
'r+' Ouvre en lecture et écriture, et place le pointeur de fichier au début du fichier.
'w' Ouvre en écriture seule ; place le pointeur de fichier au début du fichier et réduit la taille du fichier à 0. Si le fichier n'existe pas, on tente de le créer.
'w+' Ouvre en lecture et écriture ; place le pointeur de fichier au début du fichier et réduit la taille du fichier à 0. Si le fichier n'existe pas, on tente de le créer.
'a' Ouvre en écriture seule ; place le pointeur de fichier à la fin du fichier. Si le fichier n'existe pas, on tente de le créer.
'a+' Ouvre en lecture et écriture ; place le pointeur de fichier à la fin du fichier. Si le fichier n'existe pas, on tente de le créer.

Exemple

```
<?
$handle = fopen("http://www.example.com/", "r");
$handle = fopen("test/monfichier.txt", "r+");
?>
```

2- Lecture de fichier

a- fgets()

string **fgets** (resource handle , *int length*)

fgets retourne la chaîne lue jusqu'à la longueur length - 1 octet depuis le pointeur de fichier handle , ou bien la fin du fichier, ou une nouvelle ligne (qui inclue la valeur retournée), ou encore un EOF (celui qui arrive en premier). Si aucune longueur n'est fournie, la longueur par défaut est de 1 ko ou 1024 octets.

Syntaxe

```
<?
$monfichier = fopen("nom_du_fichier", "r+");
fgets ($monfichier, $taille) ;
fclose($monfichier);
?>
```

La fonction prend en paramètre :
- le handle retourné par la fonction fopen()
- la taille en octets de lecture (optionnel)

Elle retourne en sortie une chaîne de caractères.

Exemple :
L'exemple suivant va permettre de lire dans un fichier texte (fichier.txt) ligne par ligne puis d'afficher le résultat à l'écran en mettant un numéro à chaque ligne.

Fichier fichier.txt

```
Tarak Joulak
Guest0
abc123
```

Fichier page.php

```
<?
$i=0;
$fichier = 'fichier.txt';
$fp = fopen($fichier,'r');              //ouverture du fichier en lecture seulement
while (!feof($fp))                      // tant qu'on n'est pas a la fin du fichier
{
        $ligne = fgets($fp);            //Lecture ligne par ligne
        echo  'Ligne '.$i++.'--->'.$ligne.'<br>';
}
fclose($fp);
?>
```

Le résultat obtenu sera :

```
Ligne 1 ---> Tarak Joulak
Ligne 2 ---> Guest0
Ligne 3 ---> abc123
```

b -fread()

string **fread** (resource handle , int length)

Une autre manière de faire de la lecture dans un fichier est d'utiliser **fread()**
fread lit jusqu'à length octets dans le fichier référencé par handle . La lecture s'arrête lorsque length octets ont été lus, ou que l'on a atteint la fin du fichier, ou qu'une erreur survient (le premier des trois).

Syntaxe

```
<?
$monfichier = fopen("nom_du_fichier", "r+");
fread ($monfichier, $taille) ;
fclose($monfichier);
?>
```

Exemple Fichier page.php

```
<?
// Lit un fichier, et le place dans une chaîne
$filename = "fichier.txt";
$handle = fopen ($filename, "r");
$contents = fread ($handle, filesize ($filename));
echo "-->".$contents;
fclose ($handle);
?>
```

Le résultat obtenu sera :
--> Tarak Joulak Guest0 abc123

c- file()

array **file** (string filename)
Encore une autre manière de lire dans un fichier est d'utiliser la fonction **file()**
Identique à readfile, hormis le fait que file retourne le fichier dans un tableau. Chaque élément du tableau correspondant à une ligne du fichier, et les retour–chariots sont placés en fin de ligne.

Syntaxe

```
<?
file ($nom_de_fichier)
?>
```

Rq : la fonction file() ne nécessite pas l'usage de open() et close().

Exemple Fichier page.php

```
<?php
$filename = "test/fichier.txt";
$fcontents = file($filename);
echo $fcontents[0]."<br>";
echo $fcontents[1]."<br>";
?>
```

Le résultat obtenu sera :
Tarak Joulak
Guest0

3- Ecriture dans fichier

La fonction pour l'écriture dans un fichier est **fputs()**

int **fputs** (int handle , string string , int length)
fputs écrit le contenu de la chaîne string dans le fichier pointé par handle . Si la longueur length est fournie, l'écriture s'arrêtera après length octets, ou à la fin de la chaîne (le premier des deux).
fwrite retourne le nombre d'octets écrits ou FALSE en cas d'erreur.

Syntaxe

```
<?
$monfichier = fopen("nom_du_fichier", "r+");
fputs ($monfichier, "Texte à écrire");
fclose ($monfichier);
?>
```

Exemple

```
<?
$filename = "fichier.txt";
$monfichier = fopen ($filename, "a+");
$contenu="ceci est le texte a ecrire |r|n";
fputs($monfichier, $contenu);
fclose ($monfichier);
?>
```

Le résultat obtenu sera :
Le fichier fichier.txt aura une ligne supplémentaire contenant « ceci est le texte à écrire »

4- Fonctions diverses de traitement de fichier :

-feof ($handle) fonction qui retourne un booleen pour indiquer la fin du fichier parcouru.
Elle prend en entrée le handle retourné par la fonction fopen().

-**filesize** ($nom_du_fichier) fonction qui retourne la taille du fichier en octets.

-file_exists ($nom_du_fichier) fonction qui retourne un booleen indiquant si le fichier existe ou non.

IV- Passage et transmission de variables

1-Passage et transmission de variables par formulaire

Quand dans un site web un formulaire est rempli et envoyé, le contenu des champs saisis est transféré à la page destination sous forme de variables. Ce passage de variables ou de paramètres peut se faire de deux manières : en GET ou en POST.

Syntaxe

```
<html>
<body>
        <!—Envoi d'un formulaire en POST  -->
        <form method="post" action="destination.php">
                <input type="text" name="nom" size="12"><br>
                <input type="submit" value="OK">
        </form>

        <!—Envoi d'un formulaire en GET  -->
        <form method="get" action=" destination.php">
                …
        </form>
</body>
</html>
```

En GET les paramètres apparaissent associés à l'url sous formes de variables séparées par des & (http://localhost/destination.php?nom=Tarak&prenom=Joulak).
En POST le passage de paramètre se fait de manière invisible.

Selon que la méthode d'envoi a été du GET ou du POST la récupération du contenu des variables est faite selon une syntaxe différente :

Syntaxe

```
<?
//Dans le cas d'un envoi des paramètres en POST
$variable1=$_POST['nom_du_champ'] ;

//Dans le cas d'un envoi des paramètres en GET
$variable1=$_GET['nom_du_champ'] ;
?>
```

Exemple 1 :
Dans l'exemple suivant le fichier *formulaire.html* contient le script html permettant d'afficher un formulaire et d'envoyer les résultats de la saisie à la page *resultat.php* qui elle les affichera.

Fichier formulaire.html

```
<html>
<body>
        <form method="post" action="resultat.php">
        Nom : <input type="text" name="nom" size="12"><br>
        Prénom : <input type="text" name="prenom" size="12">
        <input type="submit" value="OK">
        </form>
</body>
</html>
```

Fichier resultat.php

```
<?
//Récupération des paramètres passés
$prenom = $_POST['prenom'];
$nom = $_POST['nom'];

//affichage des paramètres
echo "<center>Bonjour $prenom $nom</center>";
?>
```

En exécutant à travers le serveur web le fichier *formulaire.html*, en remplissant le formulaire et en cliquant sur OK, nous sommes emmenés vers la page *resultat.php* qui nous affiche une phrase composée des champs saisis dans le formulaire. Les champs saisis sont donc passé de formulaire.html vers resultat.php.

Exemple 2 :
L'exemple précédent peut tenir dans un seul fichier qui s'enverrait à lui même les paramètres

Fichier formulaire.php

```
 <?
if ($_POST['submit'] == "OK")
{
        //Le formulaire a été transmis
        //Récupération des paramètres passés
        $prenom = $_POST['prenom'];
        $nom = $_POST['nom'];

        //affichage des paramètres
        echo "<center>Bonjour $prenom $nom</center>";
        }
else
{
?>
        <!-- Affichage du formulaire de saisie  -->

        <form method="post" action="formulaire.php">
        Nom : <input type="text" name="nom" size="12"><br>
        Prénom : <input type="text" name="prenom" size="12">
        <input type="submit" name="submit"  value="OK">
        </form>
<?
}
?>
```

Remarque :
La définition de la destination du formulaire (action="formulaire.php") aurait pu ne pas être nominative mais exploiter une variable serveur PHP_SELF puisque les paramètres sont envoyées à la page elle même. Ainsi $_SERVER['PHP_SELF'] retournera naturellement formulaire.php.
Cela donnera donc :

*<form method="**post**" action="**<? echo $_SERVER['PHP_SELF']; ?>**" >*
à la place de
*<form method="**post**" action="**formulaire.php**">*

2-Passage et transmission de variables par hyperlien

Des paramètres ou variables peuvent passer d'une page source vers une page destination sans transiter par un formulaire pour leur envoi. Les hyperliens peuvent être des vecteurs de passage de paramètre.

Syntaxe

```
<!—Syntaxe d'envoi -->
<a href=destination.php ?variable1=contenu1&variable2=contenu2&...> Lien </a>
```

La récupération des paramètres dans la page destination se fait par le tableau ***$_GET***

```
<?
$variable1=$_GET['variable1'] ;
$variable2=$_GET['variable2'] ;
…
?>
```

Exemple :
Dans l'exemple suivant nous allons créer un fichier *menu.php* contenant un menu fait d'hyperliens. Chacun de ces hyperliens enverra des paramètres différents.
Ce menu sera appelé dans une page qui réagira différemment selon le paramètre envoyé.

Fichier menu.php

```
<table width="200" border="0" cellspacing="0" cellpadding="1">
  <tr>
    <td> <a href="page.php?menu=1">Menu1</a> </td>
  </tr>
  <tr>
    <td> <a href="page.php?menu=2">Menu2</a> </td>
  </tr>
  <tr>
    <td> <a href="page.php?menu=3">Menu3</a> </td>
  </tr>
</table>
```

Fichier page.php

```
<?
Include("menu.php") ;
echo "<br><br>" ;
$menu= $_GET['menu'] ;
switch ($menu)
   {
   case 1:
      echo  "Ceci est la page obtenue du Menu1" ;
      break;
   case 2:
      echo  "Ceci est la page obtenue du Menu2" ;
      break;
   case 3:
         echo  "Ceci est la page obtenue du Menu3" ;
         break;
   default:
       echo  "Ceci est la page d'accueil" ;
   }
?>
```

En exécutant à travers le serveur web page.php, la sélection de chacun des menus chargera à nouveau la page en envoyant des paramètres différents qui seront traités par la page.

3- Redirection

La fonction essentielle de la redirection consiste à sitôt que l'instruction a été trouvée de l'exécuter en redirigeant l'utilisateur vers la page spécifiée.

Header()

int **header**(« Location :string destination)

Syntaxe

```
<?
header("Location:$page_destination");
exit() ;
?>
```

Exemples

```
<?
header("Location:http://www.google.com");
exit() ;
?>
```

Le résultat obtenu sera :

Vous serez automatiquement redirectionné vers le site de google

```
<?
header("Location:menu.php");
exit() ;
?>
```

Le résultat obtenu sera :

Vous serez automatiquement redirectionné vers la page menu.php de votre répertoire web

Remarque importante

La fonction header doit être appelée avant la première balise HTML, et avant n'importe quel envoi de commande PHP. C'est une erreur très courante que de lire du code avec la fonction include et d'avoir des espaces ou des lignes vides dans ce code qui produisent un début de sortie avant que header n'ait été appelé.

VI-Variables persistantes: Cookies et Session

Les variables ont une durée de vie limitée : celle du script qui les appelle. Ainsi que nous l'avons vu dans le chapitre précédent l'unique moyen de transmettre ces variables de pages en pages consiste à effectuer un passage de paramètres (méthode GET ou POST) ce qui d'une manière générale est contraignant à plus d'un titre :
- Contraignant pour le développeur puisque il doit gérer par code ces passages de paramètres,
- Contraignant pour la sécurité si l'on ne désire pas que le client accède à certaines informations.

PHP offre un mécanisme de stockage d'informations de manière persistante. Autrement dit tout au long de la navigation à l'intérieur d'un site des variables seront accessibles sans avoir pour autant à les passer en paramètres.
Deux types de variables persistantes existent :

- Les variables persistantes côté client : les cookies
- Les variables persistantes côté serveur : la session

1- les Cookies

Les cookies sont un mécanisme d'enregistrement d'informations sur le client, et de lecture de ces informations. Ce système permet d'authentifier et de suivre les visiteurs d'un site. PHP supporte les cookies de manière transparente.

a- setcookie() : Cette fonction permet de définir un cookie qui sera envoyé avec le reste des en–têtes.

int setcookie (string nom_variable ,[string valeur_variable],[int expiration],[string chemin],[string domaine],[int securité])

nom_variable : nom de la variable a stocker
valeur_variable : Valeur de la variable a stocker
expiration : durée pour l'expiration du cookie
chemin : le chemin du répertoire ou doit être lu le cookie
domaine : le nom domaine
securité : le type d'entête (http, https)

Tous les arguments sauf *nom_variable* sont optionnels.
A l'instar de la fonction *header()*, *setcookie()* doit impérativement être appelée avant tout affichage de texte.

Syntaxe

```
<?
setcookie("variable1",$valeur1, $durée,$chemin,$domaine,$securite);
?>
```

Exemples

```
<?
$valeur= "Ceci est la valeur de la variable" ;
setcookie ("TestCookie",$value,time()+3600); /* expire dans une heure */
?>
```

b- *lire un cookie*

Syntaxe

```
<?
$HTTP_COOKIE_VARS["$nom_de_la_variable"] ;
?>
```

Exemple

```
<?
echo $HTTP_COOKIE_VARS["TestCookie"];
?>
```

Le résultat obtenu sera :
Ceci est la valeur de la variable

c- Tableau de variables dans un cookie

Il est possible d'envoyer un tableau de variables à stocker dans un cookie

Exemple

```
<?
setcookie( "tcookie[trois]", "troisième cookie" );
setcookie( "tcookie[deux]", "second cookie" );
setcookie( "tcookie[un]", "premier cookie" );
}
?>
```

Exemple de lecture d'un tableau stocké dans un cookie

```
<?
//récupération du tableau cookie dans la variable $cookie
$cookie=$HTTP_COOKIE_VARS["tcookie"];

//Vérification si la variable est vide ou non
if ( isset( $cookie ) )
{
        while( list( $name, $value ) = each( $cookie ) )
        {
        echo "$name == $value<br>\n";
        }
}
?>
```

Le résultat obtenu sera :
trois == troisième cookie
deux == deuxième cookie
un == premier cookie

d-Suppression d'un cookie

Pour supprimer un cookie envoyez un cookie avec une variable sans valeur et un délai dépassé.

Exemple

```
<?
Setcookie ("TestCookie","",time()-100);
?>
```

e- Limitation des cookies

Le problème majeur du cookie c'est que le client a le pouvoir de le refuser (en configurant spécifiquement son browser). Votre application risque donc de ne pas pouvoir fonctionner.
Il y a aussi des risques plus graves quant à la sécurité. L'usurpation d'identité, car ce fichier peut être recopié facilement sur un autre ordinateur et modifié puisque ce n'est qu'un fichier texte.

2- les sessions

La gestion des sessions avec PHP est un moyen de sauver des informations entre deux accès. Cela permet notamment de construire des applications personnalisées, et d'accroître l'attrait de votre site.
Chaque visiteur qui accède à votre site se voit assigner un numéro d'identifiant, appelé plus loin "identifiant de session". Celui–ci est enregistré soit dans un cookie, chez le client, soit dans l'URL.
Les sessions vous permettront d'enregistrer des variables pour les préserver et les réutiliser tout au long de la visites de votre site. Lorsqu'un visiteur accède à votre site, PHP vérifiera si une session a déjà été ouverte. Si une telle session existe déjà, l'environnement précédent sera recréé.
L'inconvénient précédemment évoqué concernant les cookies est dépassé dans la mesure ou tout est stocké sur le serveur même.

a- session_start()

session_start() permet de démarrer une session pour le client .

Syntaxe

```
<?
session_start() ;
?>
```

Cette commande doit figurer dans toutes les pages elle perpétue le transfert des variables de session au cours de la navigation dans le site.
Le compilateur PHP va alors créer alors dans le répertoire de sauvegarde des sessions, un fichier dont le nom commence par sess_ et se termine par un identifiant généré de manière aléatoire.
L'identifiant de session peut être affiché par la commande session_id(). Vous pouvez également gérer vous-même ce nom de session en utilisant session_name() avant le démarrage de la session.
La durée de vie d'une session est paramétrée par session.cache_expire

La session est perdue définitivement pour l'utilisateur lorsque :
1- Il n'a exécuté aucune action (POST ou GET) au delà de la durée de vie définie .
2- Il ferme son navigateur.
3- La commande session_destroy est appelée.

b- Ajouter des variables dans la session

L'ajout de variable dans l'environnement de session se fait au travers d'une affectation classique.
Toutefois la variable session définie doit être appelée vie la tableau ***$_SESSION[]***

Syntaxe

```
<?
session_start() ;
$_SESSION['nom_variable'] = $valeur;
?>
```

Exemple

```
<?
session_start() ;
$_SESSION['ville'] = "Tunis";
?>
```

Dans cet exemple une variable du non de *ville* contenant la valeur *Tunis* a été enregistrée dans la session courante.

c- Lire une variable dans la session

Syntaxe

```
<?
session_start() ;
$variable = $_SESSION['nom_variable'] ;
?>
```

Exemple

```
<?
session_start() ;
echo  ' Le contenu de la variable VILLE de la session est : '. $_SESSION['ville'] ;
?>

Le résultat obtenu sera :
Le contenu de la variable VILLE de la session est : Tunis
```

d- Supprimer une variable de la session

Syntaxe

```
<?
session_start() ;
unset ($_SESSION['nom_de_le_variable']);
?>
```

Exemple

```
<?
session_start() ;
unset ($_SESSION['ville']);
//Verificatin de la suppression
if ( isset ($_SESSION['ville']))
{
        $resultat = "La suppression a échoué .";
}
else
{
        $resultat = "La ville a été effacée.";
}
echo $resultat;
?>

Le résultat obtenu sera :
La ville a été effacée.
```

e- Supprimer l'environnement de session

Dans l'exemple précédent nous avons vu comment supprimer une variable de l'environnement de session.

Il reste toutefois possible de supprimer tout un environnement de session donné. Pour cela il suffit de vider le tableau global des session en le réinitialisant.

Syntaxe

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

VIII- Utilisation d'une base de données MySql

1- Introduction

PHP fonctionne nativement avec une base de données MYSQL.
MYSQL est un système de gestion de base de données (SGBD) qui permet d'entreposer des données de manière structurée (Base, Tables, Champs, Enregistrements). Le noyau de ce système permet d'accéder à l'information entreposée via un langage spécifique le SQL.
Ainsi , dans les chapitres précédents nous avions vu que l'information au mieux pouvait être stockée dans des fichier accessibles en lecture et écriture par des scripts PHP.
MYSQL vient ajouter une couche supplémentaire de stockage des données qui est plus commode, rapide et puissante d'utilisation.

Schéma de liaison entre apache/PHP/MYSQL

Voici ce qu'il peut se passer lorsque le serveur reçoit une demande d'un client de consultation d'une page en PHP qui fait appel à des données stockées sous MYSQL:

1. Le serveur WEB envoie le nom de la page PHP demandée à l'interpréteur PHP.
2. PHP exécute le script existant dans la page. Sitôt que des instructions relatives à la connexion à une base de données trouvées, PHP se charge d'envoyer les requêtes d'execution à MYSQL.
3. MySQL exécute la requête et renvoie à PHP le jeu de données résultat.
4. PHP termine son traitement et renvoie la page HTML générée au serveur web qui la transmet à l'internaute.

2- Concepts fondamentaux : Bases, tables, champs, enregistrements.

Une base de donnée correspond donc à un ensemble de données sauvegardés de manière structurée. La structuration de ses données est hiérarchisée.

BASE DE DONNEES BIBLIOTHEQUE

TABLE LIVRE

ISBN	TITRE	AUTEUR

TABLE PERIODIQUES

Une base de données peut regrouper plusieurs tables.
Ainsi une base de données BIBLIOTHEQUE par exemple contiendra une table PERIODIQUE pour stocker les périodiques, une table LIVRE pour stocker les livres.
La table LIVRE regroupera plusieurs champs destinés à décrire et qualifier les livres.
Ainsi cette table intègrera un champs NISBN, un champs TITRE et un champs AUTEUR par exemple.
La consultation de ces données se fait au travers du langage SQL.

3- Administration de MYSQL

EasyPHP installe un produit pour l'administration d'une base de données MYSQL appelé PHPMYADMIN. Ce produit est intégralement développé en PHP. Toutefois nous lui préférons MySQL-Front qui reste plus simple d'utilisation.
MYSQL-Front est une application qui permet d'administrer une base de données MYSQL à distance (Création de base, de tables, d'enregistrements, exécution de requêtes, gestion des comptes, …)

4- SQL : petit récapitulatif du langage

Structured **Q**uery **L**angage est langage standard pour communiquer avec une base de données. Il permet la création de bases, la définition de tables, la consultation des enregistrement ainsi que leur mise à jour.

a-Création d'une base

CREATE DATABASE exercice
Cette requête SQL va permettre de créer une base vide du nom de exercice.

b-Création d'une table

```
CREATE TABLE `t_personne` (
`id` INT (6) UNSIGNED AUTO_INCREMENT,
`Nom` VARCHAR (50),
`Prenom` VARCHAR (50),
`Age` TINYINT (3) UNSIGNED,
UNIQUE(`id`),
 INDEX(`id`)
)
```

Cette requête va permettre de créer une table du nom de 't_personne' contenant 4 champs :
id : un identifiant entier qui s'auto incrémentera à chaque nouvel ajout d'enregistrement
Nom : une chaîne de caractères d'au maximum 50 caractères
Prenom : une chaîne de caractères d'au maximum 50 caractères
Age : un entier

c-Ajout d'un enregistrement

INSERT INTO t_personne (id, Nom, Prenom, Age) **VALUES** (NULL, 'Joulak', 'Tarak', 32)
Cette requête SQL va permettre d'ajouter une ligne d'enregistrement à la table t_personne encore vide.
Le champs Nom prendra pour valeur Joulak
Le champs Prenom prendra pour valeur Tarak
Le champs Age prendra pour valeur 32

Il est à noter que le champs **id** étant en mode auto incrémental il ne faut pas lui assigner de valeur. Le compteur s'incrémentant tout seul à chaque nouvel enregistrement.

d-Consultation d'un enregistrement

SELECT * **FROM** t_personne **WHERE** id=1
Cette requête va extraire de la base de données la ligne d'enregistrement ayant pour identifiant 1 (id=1)

e-Mise à jour d'un enregistrement

UPDATE t_personne **SET** Age= 35 **WHERE** id=1

Cette requête va mettre à jour dans la table t_personne l'enregistrement dont le champs id est égal à 1 (id=1) en modifiant le champs Age à 35.

f-Suppression d'un enregistrement

DELETE FROM t_personne **WHERE** id=1

Cette requête SQL va supprimer de la table t_personne l'enregistrement ayant pour identifiant 1 (id=1).

5- Accéder à MYSQL via PHP

Dans ce paragraphe nous allons voir comment combiner le langage SQL aux fonctions spécifiques de PHP pour exploiter le contenu d'une base de données MYSQL.

a- Connection à un serveur MYSQL

La première opération a réaliser pour accéder à MYSQL via un script PHP correspond à la connection à un serveur de base de données MYSQL.

Syntaxe

```
<?
mysql_connect("$nom_serveur_MYSQL", "$utilisateur", "$mot_de_passe");
?>
```

Exemple

```
<?
$nom_serveur_MYSQL="localhost" ;
$utilisateur="root" ;
$mot_de_passe="" ;
mysql_connect("$nom_serveur_MYSQL", "$utilisateur", "$mot_de_passe");
?>
```

La fonction **mysql_connect()** retourne un booléen ; True si la connection est possible False si elle ne l'est pas.

b- Connection à une base de données MYSQL

Suite à la connection au serveur MYSQL, il faut choisir quelle est la base du serveur MYSQL sur laquelle nous désirons nous connecter. Un serveur MYSQL pouvant contenir plusieurs bases de données.

Syntaxe

```
<?
mysql_select_db("$nom_de_la_base");
?>
```

Exemple

```
<?
$adresse_serveur_MYSQL="localhost" ;
$utilisateur="root" ;
$mot_de_passe="" ;
$nom_de_la_base="exercice" ;
mysql_connect("$adresse_serveur_MYSQL", "$utilisateur", "$mot_de_passe");
mysql_select_db("$nom_de_la_base");
?>
```

De même la fonction **mysql_select_db()** retourne un booléen ; True si la connection est réussie, False si elle a échoué.

c- Exécution d'une requête SQL via PHP

Le lancement d'une requête SQl au travers d'un script PHP se fait par le biais de la fonction mysql_query() qui prend en paramètre la requête SQL et retourne true ou false selon que la requête ait réussi ou échoué. Pour le cas particulier d'une requête avec SELECT et si la requête a réussi alors un identifiant est retourné afin d'être exploité par d'autres fonctions.

Syntaxe

```
<?
…
$res = mysql_query("$requete_sql");
?>
```

Exemple

```
<?
$adresse_serveur_MYSQL="localhost" ;
$utilisateur="root" ;
$mot_de_passe="" ;
$nom_de_la_base="exercice" ;
$requete_sql="INSERT INTO t_personne (id, Nom, Prenom, Age) VALUES (NULL,
'Joulak', 'Tarak', 32)" ;
mysql_connect("$adresse_serveur_MYSQL", "$utilisateur", "$mot_de_passe");
mysql_select_db("$nom_de_la_base");
$res = mysql_query("$requete_sql");
?>
```

Dans l'exemple précédent une ligne d'enregistrement a été ajoutée à la table t_personne contenant Tarak Joulak 32.
En remplaçant la requête de l'exemple précédent par les requêtes SQL de modification et suppression vues dans le paragraphe précédent on obtient les résultats déjà observés en exécutant directement la requête SQL.

d- Parcourir le résultat d'un SELECT

Une requête SQL de consultation peut retourner plusieurs enregistrements. De ce fait il y a besoin de pouvoir les consulter enregistrement par enregistrement et champs par champs. **mysql_fetch_array()** est la fonction PHP destinée pour ce type de traitements.
Cette fonction prend en entrée l'identifiant retourné par mysql_query(). Elle retourne un tableau contenant la ligne demandée ou false s'il n'y a plus d'enregistrement. Elle est généralement exploitée dans une boucle WHILE.

Syntaxe

```
<?
…
$data = mysql_fetch_array($resultat_de_mysql_query) ;
?>
```

Exemple

```
<?
mysql_connect("localhost" , "root" , "");
mysql_select_db("exercice");
$requete_sql="SELECT * FROM  t_personne" ;
$res = mysql_query("$requete_sql");
while ($data= mysql_fetch_array($res))
{
echo $data['Nom'].' '.$data['Prenom']. '  ' .$data['Age']. '<br>';
}
?>
```

L'exécution de ce script affichera l'ensemble du contenu de la table t_personne avec un enregistrement par ligne.

IX- L'évolution de PHP en PHP5

La nouvelle version de PHP introduit quelques changements majeurs tout en conservant une compatibilité totale avec la version antérieure. Outre le modèle Orienté Objet entièrement remanié (mais compatible) et un nuveau parseur XML, l'une des véritables nouveautés réside principalement dans l'intégration dans PHP d'un SGBD embarqué SQLite.

SQLite est un SGBD embarqué, donc compilé par défaut, ce qui signifie que le simple fait d'installer PHP vous permet de l'utiliser. Contrairement à MYSQL, il ne nécessite donc pas de processus indépendant comme MYSQL Server. SQLite est très léger et très rapide, selon le site officiel de SQLite, il serait 2 à 3 fois plus rapide que Mysql pour la plupart des opérations et parfois 10 fois plus rapide que Postgress et probablement aussi plus rapide que beaucoup de SGBD. Ceci est assez normal puisqu'il n'a pas d'architecture client-serveur, l'interface et le moteur étant en fait contenu dans le même package.

SQLite est doté d'atouts intéressants, voici une liste non-exhaustive de ceux-ci
-Possibilité de gérer des transactions
-La gestion de contraintes d'intégrité dans un futur proche
-Gestion des triggers
-Gestion des exceptions
-Possibilité d'utiliser les fonctions PHP internes directement dans les requêtes
-Possibilité d'utiliser ses propres fonctions codées en PHP dans les requêtes
-Possède beaucoup de fonctions internes

X- Webographie

1- www.nexen.net
Site de référence en français qui décrit exhaustivement l'ensemble des fonctions de PHP et de MYSQL

2- www.php.net
Site de PHP, vous pourrez y télécharger les dernières version et suivre les innovations (en anglais)

3- www-fr.mysql.com
Site officiel de MYSQL (en français)

4- www.phpmyadmin.net
Site officiel d'Apache

5- www.easyphp.org
Site en français ou vous pourrez télécharger les dernières mises à jour de Easyphp.

6- www.phpmyadmin.net/
Site officiel de PHPMyAdmin

7- php.developpez.com/cours
Site en français ou vous trouverez un ensemble de cours sur l'usage de PHP et ses fonctionalités