TEK-UP Ecole Supérieure Privée Technologie
& Ingénierie

Développement WEB

A.U. 2019-2020

Plan du cours

I. HTML

II. CSS

III. Java Script

IV. PHP

Ch. IV

PHP

Plan du chapitre 5

Plan du chapitre 5

- Fonctions mathématiques
- Opérateurs booléens
- Instructions conditionnelles
- Instructions de boucle
- Gestion des erreurs
- Chaînes de caractères
- Tableaux
- Formulaires
- Fonctions
- Dates

Plan du chapitre 5

- Fichiers
- Sessions
- L'envoi des e-mails
- Accès à une base MySQL avec PHP

Partie I :
Les bases du langage PHP

Plan du chapitre 5

Introduction

- PHP signifiait à l'origine **Personal Home Page**. Aujourd'hui, il signifie **Php Hypertext Preprocessor**.
- Un langage de script libre dédié à Internet, directement inclus dans les pages Web.
- PHP permet de créer des pages Web dynamiques et interactives via un serveur HTTP.
- La syntaxe du PHP provient de celle du langage C, du Perl et de Java.
- Différentes versions du PHP ont été développées, depuis son apparition en 1994 :
  - 1998 : version 3.0 ;
  - 1999 : version 4.0 ;
  - 2004 : version 5.0 ;
  - Aujourd'hui, on parle de PHP 5.3 ;
- L'exécution des scripts PHP est déléguée à un composant indépendant installé sur le serveur Web souvent appelé moteur (PHP5 utilise Zend Engine).

Introduction

- Les principaux avantages du PHP sont :
  - Sa facilité d'apprentissage ;
  - Sa simplicité d'écriture ;
  - Sa souplesse d'utilisation ;
  - Sa très grande richesse fonctionnelle vis-à-vis de la connexion à des bases de données ;
  - Sa compatibilité avec différents serveurs Web (Apache, Microsoft IIS, ...) ;
  - Sa disponibilité pour différentes plateformes (Linux, Windows, MAC...) ;
  - Sa gratuité (tous ses logiciels sont open source) ;
- PHP, MySQL et Apache forment le trio ultradominant sur les serveurs Web. On parle de système LAMP pour les serveurs à Linux, WAMP pour Windows et MAMP pour MAC.

Plan du chapitre 5

Intégration du code PHP

- Le code PHP est toujours incorporé dans du code HTML.
- Les pages Web contenant des scripts PHP sont enregistrées avec l'extension **.php**.
- Il est possible d'inclure autant de scripts PHP indépendants que l'on souhaite, n'importe où dans du code HTML.
- Un code PHP est délimité par :
  - **<?php** et **?>** (la méthode la plus utilisée)
  - **<?=** et **?>** (forme courte nécessitant l'activation de la directive **short open tag** dans le fichier de configuration de PHP5 (fichier php.ini)).
  - **<script language= "php">** et **</script>** (rarement utilisé)
- Les scripts PHP peuvent être écrits :
  - directement dans le code HTML ;
  - dans des fichiers externes enregistrés avec l'extension **.inc** ou **.inc.php** et qui seront incorporés par la suite dans le code HTML en fonction des besoins ;

Intégration du code PHP

- Les fonctions permettant l'inclusion d'un fichier externe dans du code PHP :

Fonction	Description
`include("nom_fichier.ext")`	Lors de son interprétation par le serveur, cette ligne est remplacée par tout le contenu du fichier précisé en paramètre, dont vous fournissez le nom et éventuellement l'adresse complète. En cas d'erreur, par exemple si le fichier n'est pas trouvé, `include()` ne génère qu'une alerte, et le script continue.
`require("nom_fichier.ext")`	A désormais un comportement identique à `include()`, à la différence près qu'en cas d'erreur, `require()` provoque une erreur fatale et met fin au script.
`include_once("nom_fichier.ext")` `require_once("nom_fichier.ext")`	Contrairement aux deux précédentes, ces fonctions ne sont pas exécutées plusieurs fois, même si elles figurent dans une boucle ou si elles ont déjà été exécutées une fois dans le code qui précède.

Intégration du code PHP

Exemple

☐ Insertion directe du code PHP :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html>

 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
  <title>Une page PHP</title>
 </head>
 <body>
  <?php
   echo "<h3> Aujourd'hui le ". date('d / M / Y H:m:s')."</h3><hr />";
   echo "<h2>Bienvenue sur le site PHP 5</h2>";
  ?>
 </body>
 </html>
```

Intégration du code PHP

Exemple

- Inclusion d'un code externe :

fichier corps.inc

```
<?php
echo "<h3> Aujourd'hui le ". date('d / M / Y H:m:s ')."</ h3><hr />";
echo "<h2>Bienvenue sur le site PHP 5</h2>";
?>
```

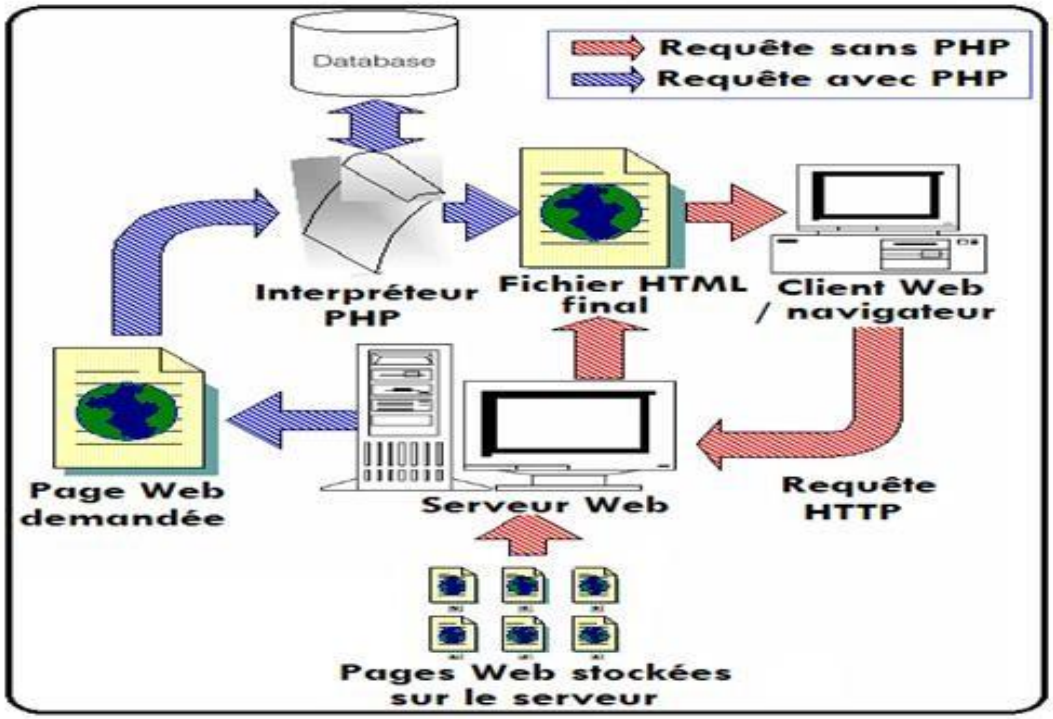
fichier principal.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html>
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
  <title>Une page PHP</title>
 </head>
 <body>
 <?php
 include("corps.inc");
 ?>
 </body>
 </html>
```

Plan du chapitre 5

Cycle de vie d'une page PHP

- Envoi d'une requête HTTP par le navigateur client vers le serveur.
  http://www.monserveur.com/codePHP.php
- Interprétation par le serveur du code PHP contenu dans la page demandée.
- Envoi par le serveur Web d'un fichier dont le contenu est purement HTML.



Plan du chapitre 5

Commentaires

- PHP supporte les trois syntaxes de commentaires suivantes :
  - Commentaires sur une seule ligne introduits par les caractères // :

```
// Ceci est un commentaire sur une seule ligne.
```

  - Commentaires sur plusieurs lignes introduits par les caractères /* et */ :

```
/* Ceci est un commentaire
   Multi-ligne. */
```

  - Commentaires de type UNIX ne comportant qu'une seule ligne introduite par le caractère # :

```
#**********************************
# Ceci est un commentaire de type UNIX.
#**********************************
```

Plan du chapitre 5

Variables

Déclaration

- Chaque variable possède un identifiant particulier commençant par le caractère $ suivi du nom de la variable.
- Les règles de création des noms de variables sont :
  - Le nom doit commencer par un caractère alphabétique ou par le caractère (_) ;
  - La longueur du nom n'est pas limitée ;
  - Le nom de la variable doit être significatif ;
  - Les variables peuvent être déclarées n'importe où dans le script à condition qu'elles soient définies avant d'être appelées ;
  - L'initialisation des variables n'est pas obligatoire et une variable non initialisée n'a pas de type précis.
  - Les noms des variables sont sensibles à la casse.

Variables

Affectation

- Le type de la variable est déterminé selon la valeur qui lui est affectée.
- On distingue deux types d'affectation pour une variable donnée :
  - **Affectation par valeur :**
    - $nomVariable=expression ;
    - **Exemple :**
      $age=85 ;
  - **Affectation par référence :**
    - $nomVariable2=&$nomVariable1 ;
    - **Exemple :**
      $employe1="Ahmed" ;
      $employe2=&$employe1 ;
    - La variable **$employe2** devient un alias de la variable **$employe1**. Ainsi, les modifications opérées sur l'une des deux variables seront répercutées sur l'autre.

Variables

Opérateurs d'affectation combinée

Opérateur	Description
+=	Addition puis affectation : `$x += $y` équivaut à `$x = $x + $y` `$y` peut être une expression complexe dont la valeur est un nombre.
-=	Soustraction puis affectation : `$x -= $y` équivaut à `$x = $x - $y` `$y` peut être une expression complexe dont la valeur est un nombre.
*=	Multiplication puis affectation : `$x *= $y` équivaut à `$x = $x * $y` `$y` peut être une expression complexe dont la valeur est un nombre.
/=	Division puis affectation : `$x /= $y` équivaut à `$x = $x / $y` `$y` peut être une expression complexe dont la valeur est un nombre différent de 0.
%=	Modulo puis affectation : `$x %= $y` équivaut à `$x = $x % $y` `$y` peut être une expression complexe dont la valeur est un nombre.
.=	Concaténation puis affectation : `$x .= $y` équivaut à `$x = $x . $y` `$y` peut être une expression littérale dont la valeur est une chaîne de caractères.

Variables

Variables prédéfinies

- PHP dispose d'un grand nombre de variables prédéfinies.
- Les variables prédéfinies stockent des informations sur le serveur et sur toutes les données pouvant transiter entre le client et le serveur Web (exp : les valeurs saisies dans un formulaire, les cookies…).
- Les variables prédéfinies se présentent sous la forme de tableaux appelés **superglobaux** qui sont accessibles en tout point de n'importe quel script.

Voir Tableau 1

Plan du chapitre 5

Constantes

Constantes personnalisées

- Une constante personnalisée est définie grâce à la fonction **define()** :

  boolean define (string nom_cte, divers valeur_cte, boolean casse) ;

  - **nom_cte :** nom de la constante ;
  - **valeur_cte :** valeur de la constante ;
  - **casse :** vaut **TRUE** si le nom de la constante est insensible à la casse et **FALSE** sinon.
  - La fonction **define()** retourne **TRUE** si la constante a bien été définie et **FALSE** en cas de problème.

- La fonction **defined(string nom_cte)** permet de vérifier l'existence d'une constante nommée **nom_cte.** Elle retourne **TRUE** si la constante existe déjà et **FALSE** sinon.

Constantes

Constantes personnalisées

Exemple :

```php
<?php
//définition insensible à la casse
define("PI",3.1415926535,TRUE);
//Utilisation
echo "La constante PI vaut ",PI,"<br />";
echo "La constante PI vaut ",pi,"<br />";
//Vérification de l'existence
if (defined( "PI")) echo "La constante PI est déjà définie","<br />";
if (defined( "pi")) echo "La constante pi est déjà définie","<br />";
//définition sensible à la casse, vérification de l'existence et utilisation
if(define("site","http://www.funhtml.com",FALSE))
{
  echo "<a href=\" " ,site, " \">Lien vers mon site </ a>";
}
?>
```

Constantes

Constantes prédéfinies

Quelques constantes prédéfinies

PHP_VERSION	Version de PHP installée sur le serveur
PHP_OS	Nom du système d'exploitation du serveur
DEFAULT_INCLUDE_PATH	Chemin d'accès aux fichiers par défaut
__FILE__	Nom du fichier en cours d'exécution
__LINE__	Numéro de la ligne en cours d'exécution

Plan du chapitre 5

Types de données

- En PHP, il n'existe pas de déclaration explicite du type d'une variable lors de sa création.
- PHP permet la manipulation d'un certain nombre de types de données différents :
  - Les types scalaires de base :
    - Entiers, avec le type **integer** : représentation des nombres entiers dans les bases 10, 8 et 16.
      Les entiers sont codés sur 32 bits. L'intervalle de valeurs des entiers est de -2^{31} à $2^{31} - 1$. Si une opération mathématique sur une variable entière l'amène à contenir une valeur en dehors de cet intervalle, elle est automatiquement convertie en type double et conserve sa nouvelle valeur.
    - Flottants, avec le type **double** ou **float** : représentation des nombres réels.
      Les réels sont codés sur 32 bits. Le type double permet de représenter l'ensemble des nombres décimaux avec une précision de 14 chiffres.
    - Chaînes de caractères, avec le type **string**.
    - Booléens, avec le type **boolean** : contient les valeurs de vérité **TRUE** ou **FALSE**.

Types de données

- Les types composés :
  - Tableaux, avec le type **array**.
  - Objets, avec le type **object**.
- Les types spéciaux :
  - Type **resource** :
    - Représente une référence à des informations présentes sur le serveur.
    - Il est le type retourné par certaines fonctions particulières.

      Exemple : les fonctions utilisées pour accéder à une base de données lors de la connexion, qui retournent une valeur de type resource permettant d'identifier chaque connexion initiée par un utilisateur afin d'être utilisée pour retourner les données après interrogation de la base par l'utilisateur concerné.
  - Type **NULL** :
    - Le type NULL (ou null) est attribué à une variable qui n'a pas de contenu ou qui a été explicitement initialisée avec la valeur NULL.
    - **N.B :** Une variable contenant une chaîne vide ou la valeur "0" n'a pas le type NULL mais string. De même, une variable contenant la valeur 0 est du type integer.

Plan du chapitre 5

Détermination du type d'une variable

string gettype ($nom_variable)	Elle permet de déterminer le type d'une variable. Elle retourne une chaîne de caractères contenant le type de la variable.
boolean is_integer($nom_variable) **ou is_int($nom_variable)**	Elle retourne TRUE si la variable est un entier, FALSE sinon.
boolean is_double($nom_variable)	Elle retourne TRUE si la variable est un double, FALSE sinon.
boolean is_string($nom_variable)	Elle retourne TRUE si la variable est une chaîne de caractères, FALSE sinon.
boolean is_bool($nom_variable)	Elle retourne TRUE si la variable est un booléen, FALSE sinon.
boolean is_array($nom_variable)	Elle retourne TRUE si la variable est un tableau, FALSE sinon.
boolean is_object($nom_variable)	Elle retourne TRUE si la variable est un objet, FALSE sinon.
boolean is_resource($nom_variable)	Elle retourne TRUE si la variable est de type resource, FALSE sinon.
boolean is_null($nom_variable)	Elle retourne TRUE si la variable est de type null, FALSE sinon.

Plan du chapitre 5

Conversion de type

- Convertir une variable d'un type dans un autre :

$resultat = (type_désiré) $nom_variable

- **Exemple :**

```php
<?php
$var="3.52 kilomètres";
$var2 = (double) $var;
echo "\$var2= ".$var2."<br />";//affiche "$var2=3.52"
$var3 = (integer) $var2;
echo "\$var3= ".$var3."<br />";//affiche "$var3=3"
?>
```

- Modifier le type de la variable elle-même :

Boolean settype ($nom_variable, "type_désiré")

Elle retourne TRUE si l'opération est réalisée et FALSE dans le cas contraire.

- **Exemple :**

```php
<?php
$var="3.52 kilomètres";
settype($var,"double");
echo "\$var= ".$var."<br />";//affiche "$var=3.52"
settype($var,"integer");
echo "\$var= ".$var."<br />";//affiche "$var=3"
?>
```

Plan du chapitre 5

Contrôle de l'état d'une variable

boolean isset($nom_variable)	Retourne la valeur FALSE si la variable n'est pas initialisée ou a la valeur NULL, et la valeur TRUE si elle a une valeur quelconque.
boolean empty($nom_variable)	Retourne la valeur TRUE si la variable n'est pas initialisée, a la valeur 0 ou NULL ou la chaîne ''0'', et la valeur FALSE si elle a une valeur quelconque.

Exemple :

```
<?php
$a=null;
if(isset($a)){echo "\$a existe déjà<br />";}
else {echo "\$a n'existe pas<br />";}
if(empty($a)){echo "\$a est vide <br />";}
else {echo "\$a a la valeur $a<br />";}
//Affiche "$a n'existe pas" et "$a est vide"
$b=0;
if(isset($b)){echo "\$b existe déjà<br />";}
else {echo "\$b n'existe pas<br />";}
if(empty($b)){echo "\$b est vide <br />";}
else {echo "\$b a la valeur $b<br />";}
//Affiche "$b existe déjà" et "$b est vide"
$c=1;
if(isset($c)){echo "\$c existe déjà<br />";}
else {echo "\$c n'existe pas<br />";}
if(empty($c)){echo "\$b est vide <br />";}
else {echo "\$c a la valeur $c<br />";}
//Affiche "$c existe déjà" et "$c a la valeur 1"
?>
```

Plan du chapitre 5

Opérateurs numériques

Opérateur	Description
+	Addition
-	Soustraction
*	Multiplication
/	Division

%	Modulo : reste de la division du premier opérande par le deuxième. Fonctionne aussi avec des opérandes décimaux. Dans ce cas, PHP ne tient compte que des parties entières de chacun des opérandes.
	`$var = 159;`
	`echo $var%7; //affiche 5 car 159=22x7 + 5.`
	`$var = 10.5;`
	`echo $var%3.5;  //affiche 1et non pas 0.`

--	Décrémentation : soustrait une unité à la variable. Il existe deux possibilités, la prédécrémentation, qui soustrait avant d'utiliser la variable, et la postdécrémentation, qui soustrait après avoir utilisé la variable.
	`$var=56;`
	`echo $var--; //affiche 56 puis décrémente $var.`
	`echo $var; //affiche 55.`
	`echo --$var; //décrémente $var puis affiche 54.`

++	Incrémentation : ajoute une unité à la variable. Il existe deux possibilités, la préincrémentation, qui ajoute 1 avant d'utiliser la variable, et la postincrémentation, qui ajoute 1 après avoir utilisé la variable.
	`$var=56;`
	`echo $var++; //affiche 56 puis incrémente $var.`
	`echo $var; //affiche 57.`
	`echo ++$var; //incrémente $var puis affiche 58.`

Plan du chapitre 5

Fonctions mathématiques

N.B : les noms des fonctions ne sont pas sensibles à la casse.

Voir Tableau 2

Plan du chapitre 5

Opérateurs booléens

- Les opérateurs booléens servent à écrire des expressions simples ou complexes qui sont évaluées par une valeur booléenne **TRUE** ou **FALSE.**
- On distingue deux types d'opérateurs booléens :
  - Opérateurs de comparaison ;

Voir Tableau 3

  - Opérateurs logiques ;

Voir Tableau 4

Plan du chapitre 5

Instructions conditionnelles

Instruction if :

```
if(expression)
{
        //bloc d'instructions ;
}
```

```
<?php
$a=6;
if(is_integer($a) && ($a<10 && $a>5) && ($a%2==0) )
{
    echo "Conditions satisfaites";
}
?>
```

Instruction if... else :

```
if(expression1)
{
    //Bloc 1
}
elseif(expression2)
{
   //Bloc 2
}
else
{
   //Bloc 3
}
```

```
<?php
$prix=55;
if($prix>100)
{
    echo "la remise est de 10 %";
}
else
{
        echo "la remise est de 5 %";
}
?>
```

Instructions conditionnelles

Opérateur ?

$var = expression ? valeur1 : valeur2	$var = ($prix>100)? "la remise est de 10 %":"la remise est de 5 %";

Instruction switch... case :

```
switch(expression)
{
   case valeur1:
   //bloc d'instructions 1;
   break;
   case valeur2:
   //bloc d'instructions 2;
   break;
   ……..
   case valeurN:
   //bloc d'instructions N;
   break;
   default:
    //bloc d'instructions par défaut;
    break;
}
```

```
<?php
$dept=75;
switch($dept)
{
    //Premier cas
    case 75:
    echo "Paris";
    break;
    //Deuxième cas
    case 78:
    echo "Hauts de Seine";
    break;
    //Troisième cas
    case 93:
    echo "Seine Saint Denis";
    break;
     //Cas par défaut
     default:
     echo "Département inconnu en Ile de   France";
     break;
}
?>
```

Plan du chapitre 5

Instructions de boucle

Boucle for :

```
for(expression1; expression2; expression3)
{
        //instruction ou bloc;
}
```

```
<?php
for($i=1;$i<7;$i++)
{
    echo "<h$i> $i :Titre de niveau $i </h$i>";
}
?>
```

Boucle while :

```
while(expression)
{
     //Bloc d'instructions à répéter;
}
```

```
<?php
$n=1;
while($n%7!=0 )
{
   $n = rand(1,100);
   echo $n,"  /";
}
?>
```

Boucle do... while :

```
do
{
   //bloc d'instructions ;
}
while(expression);
```

```
<?php
do
{
     $n = rand(1,100);
     echo $n,"  / ";
}
while($n%7!=0);
?>
```

Instructions de boucle

Boucle foreach :

```
foreach($tableau as $valeur)
{
    //bloc utilisant la valeur de l'élément courant
}
```

```
<?php
//Création du tableau de 9 éléments
for($i=0;$i<=8;$i++)
{
    $tab[$i] = pow(2,$i);
}
//Lecture des valeurs du tableau
echo"Les puissances de 2 sont :";
foreach($tab as $val)
{
    echo $val." : ";
}
?>

/* résultat affiché :
Les puissances de 2 sont : 1 : 2 : 4 : 8 : 16 :
32 : 64 : 128 : 256:
*/
```

Instructions de boucle

Boucle foreach :

```
foreach($tableau as $indice=>$valeur)
{
    //bloc utilisant la valeur et l'indice de l'élément courant
}
```

```
<?php
//Création du tableau
for($i=0;$i<=8;$i++)
{
    $tab[$i] = pow(2,$i);
}
//Lecture des indices et des valeurs
foreach($tab as $ind=>$val)
{
   echo " 2 puissance $ind vaut $val <br />";
}
?>

/* résultat affiché :
2 puissance 0 vaut 1
2 puissance 1 vaut 2
.....
2 puissance 7 vaut 128
2 puissance 8 vaut 256
*/
```

Instructions de boucle

Sortie anticipée des boucles

- **Instruction break :** permet d'arrêter complètement une boucle **for, foreach** ou **while** avant son terme normal si une condition particulière est vérifiée.

```
<?php
    //Création d'un tableau de noms
    $tab[1]="Basile";
    $tab[2]="Conan";
    $tab[3]="Albert";
    $tab[4]="Vincent";
    //Boucle de lecture du tableau
    for($i=1;$i<count($tab);$i++)
    {
        if ($tab[$i][0]=="A")
        {
            echo "Le premier nom commençant par A est le n° $i: ",$tab[$i];
            break;
        }
    }
?>
```

Instructions de boucle

Sortie anticipée des boucles

- **Instruction continue :** n'arrête pas la boucle en cours mais seulement l'itération en cours. La variable compteur est incrémentée immédiatement, et toutes les instructions qui suivent le mot-clé **continue** ne sont pas exécutées lors de l'itération en cours.

```php
<?php
    //Intérruption d'une boucle for
    for($i=0;$i<20;$i++)
    {
        if($i%5==0)
        {
            continue;
        }
        echo $i,"<br />";
    }
?>
```

Plan du chapitre 5

Gestion des erreurs

- Il existe différents types d'erreurs :
  - Erreur de syntaxe lors de la compilation du programme ;
  - Erreur d'exécution lors de l'exécution du programme ;
  - Erreur de données : donnée inattendue, incompatible avec les routines développées pour son traitement (lettre dans un nombre représentant une quantité, ...) ;
- Lorsqu'une erreur est rencontrée dans un script PHP :
  - un message s'affiche indiquant la nature de l'erreur, sa cause, le nom du fichier de script et la ligne du script où l'erreur s'est produite ;
  - selon l'erreur, l'exécution du script se termine à l'endroit de l'erreur, ou simplement une ligne du script ne s'exécute pas, ou toutes les lignes peuvent néanmoins s'exécuter.

Gestion des erreurs

- Les messages d'erreur affichés sont de trois types :
  - Notices: Erreurs non critiques, par défaut non affichées. Toutes les instructions ont néanmoins pu être exécutées ;

```php
<?php
error_reporting(E_ALL);
$message_2 = "c'est la rose";
$message = $message_1.$message_2 ;
echo $message ;
?>
```

Notice: Undefined variable: message_1 in **/homez.60/poulhes/atelierphp_net/exercises/lesson_11_a0_notice.php** on line **4**
c'est la rose

  - Warnings: Une instruction n'a pu être correctement exécutée (exp: fichier manquant...), néanmoins le script peut continuer son exécution ;

```php
<?php
$a=10; $b=0; echo $a/$b;
?>
```

Warning: Division by zero in **c:\wamp5\www\php5\c3instructions\instruct3.15a.php** on line **4**

  - Fatal errors: Erreurs fatales : le script s'arrête : erreur de syntaxe...

```php
<?php
echo "-- begin --";
// Appel d'une fonction non existante
// qui va générer une E_ERROR (fatale)
someFunction();
// La ligne suivante ne sera jamais executée
echo "-- end --";
function somefonction()
{
   echo "Ce message est à l'intérieur de la fonction";
}
?>
```

-- begin --
Fatal error: Call to undefined function: somefunction() in **/homez.60/poulhes/atelierphp_net/exercises/lesson_11_b_fatal.php** on line **5**

Gestion des erreurs

- Le but de la gestion des erreurs consiste à signaler « proprement » les problèmes au visiteur afin d'éviter l'affichage des messages d'erreur bruts tels que envoyés par PHP au navigateur.

Gestion des erreurs

Suppression des messages d'erreur

- Deux méthodes pour éviter l'affichage des messages d'erreur de PHP dans le navigateur, à savoir :
  - Faire précéder l'appel d'une fonction du caractère @ en écrivant.
    - **Exemple :** @fopen ("fichier.txt","r").
  - Utiliser la fonction error_reporting(), définit le niveau d'erreur pour lequel le serveur doit renvoyer une erreur.
    - **Syntaxe :** int error_reporting ([int niveau]).
    - Le paramètre **niveau** permet de choisir le niveau d'affichage des messages d'erreur. Ses valeurs possibles sont :

Constante	Valeur	Niveau d'affichage
E_ERROR	1	Erreur fatale qui provoque l'arrêt du script, par exemple, l'appel d'une fonction qui n'existe pas.
E_WARNING	2	Avertissement ne provoquant pas l'arrêt du script, par exemple, une division par 0.
E_PARSE	4	Erreur de syntaxe détectée par l'analyseur PHP et provoquant l'arrêt du script, par exemple l'oubli du point-virgule en fin de ligne.
E_NOTICE	8	Avis que le script a rencontré un problème simple qui peut ne pas être une erreur.
E_ALL	4095	Toutes les erreurs

Gestion des erreurs

Suppression des messages d'erreur

☐ **Exemple :**

```
<?php
// set on ne veut voir que les erreurs notice
error_reporting(8);
//une erreur notice, variable non définie
echo $myVar ;
$string = "L'important, c'est la rose";
/*On sait que l'instruction suivante génère un warning car sting n'est pas
un tableau*/
print (join('',$string));
/*mais comme ce n'est pas une erreur fatale, l'instruction suivante est
exécutée */
echo "<br/>-- end --<br/>";
?>
```

```
Notice:Undefined variable:
myVarin
/homez.60/poulhes/atelier
php_net/exercises/lesson_
11_c0_error_reporting_fun
ction.php on line 5

-- end --
```

☐ **N.B:** error_reporting(0); // Empêche tout affichage d'erreur

Plan du chapitre 5

Chaînes de caractères

Définition d'une chaîne de caractères

- Une chaîne de caractères est une suite de caractères alphanumériques contenus entre des guillemets simples ou doubles.

```
$chaine='Bonjour tout le monde' ;
Ou
$chaine="Bonjour tout le monde" ;
```

- Si une chaîne contient une variable, celle-ci est évaluée et sa valeur est incorporée dans la chaîne uniquement si les guillemets doubles sont utilisés.

```
$chaine1='Bonjour' ;
$chaine2=" $chaine1 tout le monde" ;  // affiche : Bonjour tout le monde
$chaine3=' $chaine1 tout le monde' ;  // affiche : $chaine1 tout le monde
```

Chaînes de caractères

Séquences d'échappement

Séquence	Signification
\'	Affiche une apostrophe.
\"	Affiche des guillemets.
\$	Affiche le signe $.
\\	Affiche un antislash.
\n	Nouvelle ligne (code ASCII 0x0A).
\r	Retour chariot (code ASCII 0x0D).
\t	Tabulation horizontale (code ASCII 0x09).
\[0-7] {1,3}	Séquence de caractères désignant un nombre octal (de 1 à 3 caractères 0 à 7) et affichant le caractère correspondant : `echo "\115\171\123\121\114"; //Affiche MySQL.`
\x[0-9 A-F a-f] {1,2}	Séquence de caractères désignant un nombre hexadécimal (de 1 à 2 caractères 0 à 9 et A à F ou a à f) et affichant le caractère correspondant : `echo "\x4D\x79\x53\x51\x4C"; //Affiche MySQL.`

Chaînes de caractères

Séquences d'échappement

Fonctions	Exemples
string addslashes (string $ch)	$ch="Le répertoire est : 'C:\PHP_doc\php5'"; $ch = addslashes($ch); echo $ch; $ch =stripslashes($ch); echo $ch; /* Affiche : Le répertoire est : \'C:\\PHP_doc\\php5\' Le répertoire est : 'C:\PHP_doc\php5' */
string stripslashes (string $ch)	

Chaînes de caractères

Concaténation des chaînes

- L'opérateur PHP de concaténation est le point (.), qui fusionne deux chaînes littérales ou contenues dans des variables en une seule chaîne.
- **Exemple :**

```
$chaine=''Bonjour''.'' tout le monde'';
echo $chaine ; // affiche : Bonjour tout le monde
ou
print ($chaine) ; // affiche : Bonjour tout le monde
```

```
$chaine1='Bonjour' ;
$chaine2='' tout le monde'' ;
$chaine3=$chaine1. $chaine2;
echo $chaine3 ;  // affiche : Bonjour tout le monde
ou
print ($chaine3) ; // affiche : Bonjour tout le monde
```

```
$chaine1='Bonjour' ;
echo  $chaine1.'' tout le monde'' ; // affiche : Bonjour tout le monde
ou
echo  $chaine1,'' tout le monde'' ; // affiche : Bonjour tout le monde
ou
print ($chaine1.'' tout le monde'') ; // affiche : Bonjour tout le monde
```

Chaînes de caractères

Longueur d'une chaîne et codes des caractères

int strlen(string $chaine)	Détermine le nombre de caractères d'une chaîne.
int ord(string caractere)	Retrouve le code d'un caractère.
string chr(int code)	Retrouve le caractère à partir de son code.

Chaînes de caractères

Mise en forme des chaînes

- Les principales fonctions de mise en forme des chaînes en PHP sont :

Voir Tableau 6

- Exemples :

Voir Tableau 7

Chaînes de caractères

Recherche de sous-chaînes

- Les principales fonctions permettant la recherche de sous-chaînes en PHP sont :

Voir Tableau 8

- **Exemples :**

Voir Tableau 9

Chaînes de caractères

Comparaison de chaînes

- Les opérateurs de comparaison usuels sont utilisables avec les chaînes.

Voir Tableau 10

- Les principales fonctions permettant la comparaison des chaînes sont :

Voir Tableau 11

Chaînes de caractères

Transformation de chaînes en tableaux

- Les principales fonctions permettant la transformation de chaînes en tableaux sont :

Voir Tableau 12

Plan du chapitre 5

Tableaux

Création d'un tableau

- La fonction permettant de créer des tableaux est la fonction **array()**.
- On distingue deux types de tableaux :
  - **Les tableaux indicés :**

```
$tab[n] = valeur;
$tab = array(valeur0,valeur1,...,valeurN) ;
```

N.B :

- Le premier élément d'un tableau indicé est repéré par l'indice 0.
- Les éléments d'un tableau peuvent appartenir à des types distincts.
- Les éléments d'un tableau peuvent avoir des indices négatifs. Un indice négatif permet d'accéder aux éléments à partir de la fin du tableau en comptant à rebours. Le dernier élément du tableau non vide est toujours $tab [-1].

  - **Les tableaux associatifs :**

```
$tabasso = array("cléA"=>valeurA, "cléB"=>valeurB,... "cléZ"=>valeurZ);
```

N.B : Dans un tableau associatif, la notion d'ordre des éléments perd la valeur qu'elle peut avoir dans un tableau indicé.

Tableaux

Création d'un tableau multidimensionnel

Script PHP :

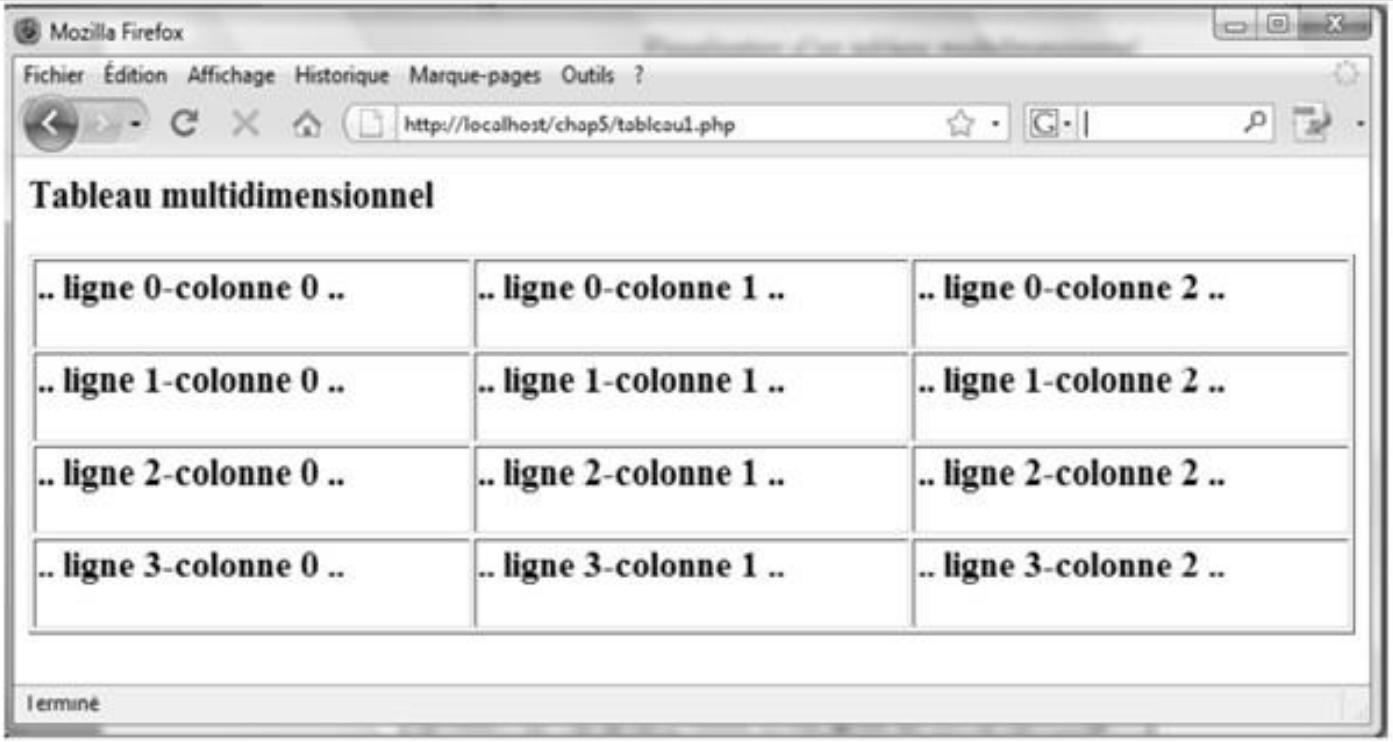
```
<? php
        $tabmulti=array(array("ligne 0-colonne 0","ligne 0-colonne
        1","ligne 0-colonne 2"),
        array("ligne 1-colonne 0","ligne 1-colonne 1","ligne 1-colonne 2"),
        array("ligne 2-colonne 0","ligne 2-colonne 1","ligne 2-colonne 2"),
        array("ligne 3-colonne 0","ligne 3-colonne 1","ligne 3-colonne 2"));

        echo "<h3>Tableau multidimensionnel</h3><table  border='1'
        width=\"100% \"> <tboby>";
        for ($i=0;$i<count($tabmulti);$i++)
        {
            echo "<tr>";
            for($j=0;$j<count($tabmulti[$i]);$j++)
            {
                echo "<td><h3> .. ",$tabmulti[$i][$j]," .. </h3></td>";
            }
            echo "</tr>";
        }
        echo " </tbody> </table> ";
?>
```

Tableaux

Création d'un tableau multidimensionnel

Résultat affiché :



Tableaux

Détermination du nombre d'éléments dans un tableau

- La fonction **count()** permet de déterminer le nombre d'éléments d'un tableau.
- **Exemples :**

Exemple 1

```
<?php
$a[0] = 1;
$a[1] = 3;
$a[2] = 5;
$result = count($a);
// $result == 3
?>
```

Exemple 2

```
<?php
$b[0]  = 7;
$b[5]  = 9;
$b[10] = 11;
$result = count($b);
// $result == 3
?>
```

Exemple 3

```
<?php
$result = count(null);
// $result == 0

$result = count(false);
// $result == 1
?>
```

Exemple 4

```
<?php
$food = array('fruits' => array('orange', 'banana', 'apple'),
                          'veggie' => array('carrot', 'collard', 'pea'));
// recursive count
echo count($food, COUNT_RECURSIVE); // output 6
// normal count
echo count($food); // output 2
?>
```

Tableaux

Lecture et affichage des éléments d'un tableau

- Affichage avec la fonction **print_r()**.
- **Exemple :**

Script PHP :

```
<?php
$tabmulti=array(array("ligne 0-colonne 0","ligne 0-colonne 1","ligne 0-colonne 2"),
array("ligne 1-colonne 0","ligne 1-colonne 1","ligne 1-colonne 2"),
array("ligne 2-colonne 0","ligne 2-colonne 1","ligne 2-colonne 2"),
array("ligne 3-colonne 0","ligne 3-colonne 1","ligne 3-colonne 2"));
print_r($tabmulti);
?>
```

Résultat affiché :

```
Array (
[0] => Array ( [0] => ligne 0-colonne 0 [1] => ligne 0-colonne 1 [2] => ligne 0-colonne 2 )
[1] => Array ( [0] => ligne 1-colonne 0 [1] => ligne 1-colonne 1 [2] => ligne 1-colonne 2 )
[2] => Array ( [0] => ligne 2-colonne 0 [1] => ligne 2-colonne 1 [2] => ligne 2-colonne 2 )
[3] => Array ( [0] => ligne 3-colonne 0 [1] => ligne 3-colonne 1 [2] => ligne 3-colonne 2 ))
```

Tableaux

Lecture et affichage des éléments d'un tableau

- Lecture et affichage avec la boucle **for.**
- **Exemple :**

Script PHP

```
<?php
$montab=array("Paris","London","Brüssel");
for ($i=0;$i<count($montab);$i++)
{
    echo "L'élément $i est $montab[$i]<br />";
}
?>
```

Résultat affiché

```
L'élément 0 est Paris
L'élément 1 est London
L'élément 2 est Brüssel
```

Tableaux

Lecture et affichage des éléments d'un tableau

- Lecture et affichage avec la boucle **while.**
- **Exemple :**

Script PHP

```
<?php
$montab=array("Paris","London","Brüssel");
$i=0;
// isset($montab[$i]) retourne FALSE lorsque
$i dépasse le nombre d'éléments du tableau
while(isset($montab[$i]) )
{
    echo "L'élément $i est $montab[$i]<br />";
    $i++;
}
?>
```

Résultat affiché

```
L'élément 0 est Paris
L'élément 1 est London
L'élément 2 est Brüssel
```

Tableaux

Lecture et affichage des éléments d'un tableau

- Lecture et affichage avec la fonction **each()**.
  - **N.B :** la fonction **each()** est utilisée lorsque les indices des différents éléments d'un tableau donné ne sont pas consécutifs.
  - **Syntaxe :**

    $element = each($tab)
    - $element est un tableau à 4 éléments :
    - $element[0], qui contient l'indice de l'élément courant.
    - $element[1], qui contient la valeur de l'élément courant.
    - $element["key"], qui contient la clé de l'élément courant.
    - $element["value"], qui contient la valeur de l'élément courant.
  - L'expression **$element = each($tab)** retourne TRUE tant que le tableau contient des éléments.
- **Exemple :**

Voir Tableau 13

Tableaux

Lecture et affichage des éléments d'un tableau

- Lecture et affichage avec l'instruction **foreach.**
- **Exemple :**

Voir Tableau 14

Tableaux

Extraction d'une partie d'un tableau

- Il est possible de créer un nouveau tableau comme sous-ensemble du tableau initial et ne contenant qu'un nombre déterminé de ses éléments, grâce à la fonction **array_slice().**
- **Syntaxe :**

```
$sous_tab = array_slice(array  $tab,int ind, int nb)
```

Tableaux

Extraction d'une partie d'un tableau

- Si ind et nb sont positifs, le tableau $sous_tab contient nb éléments du tableau initial extraits en commençant à l'indice ind.

  Exemple :

  array_slice($tab,2,3) retourne un tableau comprenant 3 éléments extraits à partir de l'indice 2. Il contient donc les éléments d'indice 2, 3 et 4 du tableau $tab.

- Si ind est négatif et que nb est positif, le compte des éléments se fait en partant de la fin du tableau $tab (le dernier élément du tableau est affecté virtuellement de l'indice –1). Le paramètre nb désigne encore le nombre d'élément à extraire.

  Exemple :

  array_slice($tab,–5,4) retourne quatre éléments de $tab extraits en commençant au cinquième à partir de la fin.

Tableaux

Extraction d'une partie d'un tableau

- Si ind est positif et nb négatif, le tableau $sous_tab contient les éléments de $tab extraits en commençant à l'indice ind et en s'arrêtant à celui qui a l'indice négatif virtuel nb (toujours en commençant par la fin).

  Exemple :

  array_slice($tab,2,–4) retourne tous les éléments à partir de l'indice 2 jusqu'à la fin, sauf les quatre derniers.

- Si ind et nb sont négatifs, le tableau $sous_tab contient les éléments de $tab extraits en commençant à l'indice négatif ind et en s'arrêtant à celui d'indice négatif nb.

  Exemple :

  array_slice($tab,–5,–2) retourne trois éléments compris entre les indices virtuels –5 compris et –2 non compris.

Tableaux

Ajout et suppression d'éléments dans un tableau	
int array_push($tab, valeur1,..., valeurN)	ajoute en une seule opération les N éléments passés en paramètres à la fin du tableau désigné par la variable $tab.
int array_unshift($tab, valeur1,..., valeurN)	ajoute au tableau $tab les *N* éléments passés en paramètres au début du tableau.
array_pop($tab)	supprime le dernier élément du tableau $tab et retourne cet élément s'il existe ou la valeur NULL dans le cas contraire.
array_shift($tab)	supprime le premier élément du tableau $tab et retourne cet élément s'il existe ou la valeur NULL dans le cas contraire.
unset(element_a_supprimer)	supprime un élément d'indice ou de clé quelconque du tableau $tab. Cette fonction n'a pas d'effet sur les autres indices du tableau, qui conservent tous la valeur qu'ils avaient avant la suppression.

Tableaux

Ajout et suppression d'éléments dans un tableau

Exemple :

Voir Tableau 15

Tableaux

Élimination des éléments faisant double emploi dans un tableau

array array_unique($tab)	retourne un nouveau tableau ne contenant que la dernière occurrence de chaque valeur présente plusieurs fois dans le tableau $tab. Les indices ou les clés associées à chaque élément sont conservés, et le tableau retourné comporte des « trous » dans la suite des indices si ces derniers sont numériques.

Exemple :

Script PHP :

```
<?php
$tab = array("Jacques","Paul","Pierre","Alban","Paul","Jack","Paul");
$tab2 = array_unique($tab);
print_r($tab2);
?>
```

Résultat affiché :

```
Array ( [0] => Jacques [2] => Pierre [3] => Alban [5] => Jack [6] => Paul )
```

Tableaux

Fusion des tableaux

$tab = array_merge($tab1,$tab2,...,$tabN)
retourne dans $tab l'ensemble des éléments présents dans les tableaux $tab1, $tab2, ..., $tabN. Si les tableaux à fusionner sont indicés, les éléments du tableau passé en premier paramètre sont conservés, ceux des autres paramètres ayant les indices suivants. Les éléments présents dans plusieurs des paramètres sont présents en double dans le tableau final. Si les tableaux à fusionner sont associatifs, les clés et les associations clé-valeur sont préservées. Par contre, si plusieurs des paramètres ont des clés communes, seule l'association clé-valeur du dernier paramètre est conservée, et celle du tableau précédent est perdue.

$tab = array_merge_recursive()
Cette fonction n'efface pas la première valeur associée à une clé double mais associe à chaque clé présente plusieurs fois un tableau indicé contenant toutes les valeurs ayant la même clé.

Tableaux

Intersection et différence de deux tableaux

Fonction	Description
array array_intersect($tab1,$tab2)	Cette fonction retourne un tableau contenant tous les éléments communs aux tableaux $tab1 et $tab2. Les indices associés aux valeurs du tableau retourné comme résultat correspondent à ceux du tableau passé en premier paramètre.
array array_diff($tab1,$tab2)	Elle retourne un tableau contenant les éléments présents dans le premier paramètre mais pas dans le second. Les indices associés aux valeurs dans les tableaux d'origine sont conservés.

Tableaux

Intersection et différence de deux tableaux

Exemple :

Voir Tableau 16

Tableaux

Tri d'un tableau indicé

Tri selon l'ordre ASCII	
array sort($tab)	Trie les valeurs du tableau $tab en ordre croissant des codes ASCII des caractères qui les composent (donc en tenant compte de la casse des caractères). Les correspondances entre les indices et les valeurs des éléments sont perdues après le tri.
array rsort($tab)	Trie les valeurs du tableau $tab en ordre décroissant des codes ASCII des caractères qui les composent. Les correspondances entre les indices et les valeurs des éléments sont perdues après le tri.
array array_reverse($tab)	Inverse l'ordre des valeurs des éléments de $tab. Les indices sont perdus.

Tableaux

Tri d'un tableau indicé

Exemple :

Voir Tableau 17

Tableaux

Tri d'un tableau indicé

Tri selon l'ordre naturel	
array natsort($tab)	Trie les valeurs du tableau $tab selon l'ordre naturel croissant des caractères qui les composent. Le tri étant effectué en tenant compte de la casse, les majuscules sont placées avant les minuscules. Les correspondances entre les indices ou les clés et les valeurs des éléments sont sauvegardées après le tri, ce qui rend la fonction également applicable aux tableaux associatifs.
array natcasesort($tab)	Trie les valeurs du tableau $tab selon l'ordre naturel croissant, sans tenir compte de la casse, ce qui correspond davantage à l'ordre courant du dictionnaire. Les correspondances entre les indices ou les clés et les valeurs des éléments sont sauvegardées après le tri.

Tableaux

Tri d'un tableau indicé

Exemple :

Voir Tableau 18

Tableaux

Tri d'un tableau indicé

N.B : Il est déconseillé d'utiliser une boucle **for** pour lire l'ensemble des données, au risque de perdre l'ordre créé par le tri vu que les fonctions **natsort()** et **natcasesort()** conservent les correspondances entre les indices ou les clés et les valeurs. Une boucle **foreach** est indispensable, même pour des tableaux indicés.

Tableaux

Tri d'un tableau associatif

- Tri des valeurs :

Tri selon le code ASCII	
void asort(array $tab)	Trie les valeurs du tableau $tab selon l'ordre croissant des codes ASCII des caractères qui les composent en préservant les associations clé-valeur.
void arsort(array $tab)	Trie les valeurs du tableau $tab selon l'ordre décroissant des codes ASCII des caractères qui les composent en préservant les associations clé-valeur.

Tableaux

Tri d'un tableau associatif

☐ Tri des valeurs :

Tri selon l'ordre naturel	
array natsort($tab)	Trie les valeurs du tableau $tab selon l'ordre naturel croissant des caractères qui les composent. Le tri étant effectué en tenant compte de la casse, les majuscules sont placées avant les minuscules. Les correspondances entre les indices ou les clés et les valeurs des éléments sont sauvegardées après le tri, ce qui rend la fonction également applicable aux tableaux associatifs.
array natcasesort($tab)	Trie les valeurs du tableau $tab selon l'ordre naturel croissant, sans tenir compte de la casse, ce qui correspond davantage à l'ordre courant du dictionnaire. Les correspondances entre les indices ou les clés et les valeurs des éléments sont sauvegardées après le tri.

Tableaux

Tri d'un tableau associatif

☐ **Tri des clés :**

boolean ksort(array $tab)	trie les clés de $tab selon l'ordre croissant des codes ASCII des caractères. Les associations clé-valeur sont conservées. Cette fonction retourne une valeur booléenne indiquant si l'opération de tri a réussi ou non.
boolean krsort(array $tab)	trie les clés de $tab selon l'ordre décroissant des codes ASCII des caractères. Les associations clé-valeur sont conservées. Cette fonction retourne une valeur booléenne indiquant si l'opération de tri a réussi ou non.

N.B : Il est possible de transformer la casse des clés avant le tri en appliquant au tableau la fonction **array_change_key_case() :**

array array_change_key_case (array $tab, int CTE)

Cette fonction transforme toutes les clés du tableau $tab en minuscules si la constante **CTE** vaut **CASE_LOWER** (valeur par défaut) ou en majuscules si elle vaut **CASE_UPPER**.

Tableaux

Tri d'un tableau associatif

Exemple :

Voir Tableau 19

Tableaux

Sélection des éléments

array array_filter(array $tab, string "nom_fonction")
Elle retourne un nouveau tableau ne contenant que les éléments de $tab qui répondent à la condition définie dans la fonction dont le nom est passé en second paramètre. Le tableau initial est conservé.

Exemple :

Voir Tableau 20

Tableaux

Application d'une fonction aux éléments d'un tableau

int array_walk($tab,"nom_fonction")
Applique la fonction dont le nom est passé en paramètre à tous les éléments du tableau qu'il soit indicé ou associatif. La fonction appliquée aux valeurs doit être une fonction personnalisée et non une fonction native de PHP.
divers array_reduce(array $tab, string "nom_fonction")
Applique la fonction dont le nom est passé en paramètre pour retourner un seul résultat à partir de l'ensemble des valeurs contenues dans le tableau (exp : somme ou produit des éléments du tableau).

Exemple :

Voir Tableau 21

Plan du chapitre 5

Formulaires

Récupération des données d'un formulaire

- Lorsque l'utilisateur clique sur le bouton d'envoi après avoir rempli les différents champs du formulaire, une requête HTTP est envoyée au serveur à destination du script désigné par l'attribut **action** de l'élément **<form>**.

- La requête contient toutes les associations entre les noms des champs et leurs valeurs. Ces associations se trouvent dans l'en-tête HTTP si la méthode **POST** est utilisée et dans l'URL s'il s'agit de la méthode **GET**.

Formulaires

Récupération des données d'un formulaire

Valeurs uniques :

- Les valeurs uniques proviennent des champs de formulaire dans lesquels l'utilisateur ne peut entrer qu'une seule valeur (exp : un texte), ou ne peut faire qu'un seul choix (bouton radio, liste de sélection à choix unique).
- Les valeurs lues sont contenues sur le serveur dans des tableaux associatifs dits **superglobaux** appelés **$_POST** et **$_GET**, selon la méthode choisie (POST ou GET) et dont les clés sont les noms associés aux champs par l'attribut **name**.
- **Exemple 1 :** Voir Tableau 22-1
- **Exemple 2 :** Voir Tableau 22-2

Formulaires

Récupération des données d'un formulaire

Valeurs multiples :

- Pour permettre à l'utilisateur de saisir plusieurs valeurs sous un même nom de composant (exp : un groupe de cases à cocher ayant le même attribut **name**), il faut définir l'attribut **multiple** dans le composant en question.
- Dans le cas de valeurs multiples, les données sont récupérées côté serveur sous la forme d'un tableau. Il est donc nécessaire de faire suivre le nom du composant de crochets tel est le cas lors de la création d'une variable de type **array**.
- **Exemple :**

Voir Tableau 22-3

Formulaires

Transfert de fichiers vers le serveur

- Comparé au transfert de données, le transfert de fichiers présente un problème de sécurité pour les sites web puisque des fichiers vont être écrits et éventuellement exécutés sur le serveur.
- Il est ainsi incontournable d'utiliser l'attribut **accept** de l'élément **<input** /> qui détermine le type de fichiers acceptés (exp : image/gif, text/html…).
- Il est aussi possible de limiter la taille maximale des fichiers à transmettre via un champ caché nommé **MAX_FILE_SIZE** dont l'attribut **value** contient la taille maximale admise en octet.

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

Formulaires

Transfert de fichiers vers le serveur

- Dans le cas de transfert de fichiers dans un formulaire, l'élément **<form>** doit avoir l'attribut **method** à la valeur **post** et l'attribut **enctype** à la valeur **multipart/form-data.**
- Un clic sur le bouton **Submit** provoque l'envoi du fichier sélectionné au serveur et son traitement par un script. Le fichier est donc temporairement mis dans un répertoire tampon défini par la directive **"upload_tmp_dir"** du fichier **php.ini** et est enregistré sous un nom différent de celui qu'il avait sur le poste client.
- Si le fichier transféré ne subit aucun traitement, il est perdu lors de la déconnexion du client.
- **Exemple :**

Voir Tableau 23

Formulaires

Transfert de fichiers vers le serveur

- Il est possible de transférer plusieurs fichiers simultanément en utilisant la syntaxe suivante (suite de l'exemple) :

```
<input type="file" name="fich[]" accept="image/gif" size="50"/>
```

Formulaires

Gestion de boutons d'envoi multiples

- Il est possible d'avoir plusieurs boutons **submit** dans un même formulaire dont l'activation d'un d'entre eux par l'utilisateur déclenche une action différente définie par l'attribut **value.**
- **Exemple :**

Voir Tableau 24

Plan du chapitre 5

Fonctions

Déclaration d'une fonction

```
function  mafonction($x,$y,...)
{
    //code de définition de la fonction
     return $var;   /* dans le cas où la fonction retourne une valeur, ce qui
     n'est pas obligatoire */
}
```

- Depuis PHP 4, la position de la déclaration d'une fonction dans le script n'a pas d'importance. Il est ainsi possible d'appeler une fonction au début du script alors qu'elle n'est définie qu'en fin de script.
- Pour les fonctions définies dans des scripts séparés, il est préférable de les inclure dès le début du script qui les utilise via l'instruction **include()** ou **require()**.

Fonctions

Appel d'une fonction

Fonction ne retournant pas de valeur :
mafonction($variable1,$variable2,...) ; ou mafonction(valeur1, valeur2,...);
Fonction retournant une valeur :
$valeurRetournee=mafonction(valeur1, valeur2,...);

- Exemple :

Voir Tableau 25

Fonctions

Fonction retournant plusieurs valeurs

- Pour qu'une fonction en PHP puisse retourner plusieurs variables, on a recours au type **array**.
- **Exemple :**

Voir Tableau 26

Fonctions

Fonction avec paramètres par défaut

- Dans la définition d'une fonction, tous les paramètres qui ont une valeur par défaut doivent figurer en dernier dans la liste des variables.
- **Exemple :**

Voir Tableau 27

Fonctions

Fonctions avec un nombre variable de paramètres

- Il existe deux méthodes pour définir des fonctions dont le nombre de paramètres à passer n'est pas connu à l'avance :
  - Utiliser le type **array** pour les paramètres ;
  - Utiliser les fonctions particulières de PHP :

integer func_num_args()
s'utilise sans argument et seulement dans le corps même d'une fonction. Elle retourne le nombre d'arguments passés à cette fonction.
divers func_get_arg(integer $N)
retourne la valeur du paramètre passé à la position $N. Comme dans les tableaux, le premier paramètre a l'indice 0.
array func_get_args()
s'utilise sans paramètre et retourne un tableau indicé contenant tous les paramètres passés à la fonction.

Exemples : Voir Tableau 28-1

Voir Tableau 28-2

Fonctions

Portée des variables

Variables locales et globales

- Toute variable utilisée dans la déclaration d'une fonction est, sauf indication contraire, locale au bloc de définition de la fonction.
- Toute variable définie en dehors d'une fonction ou d'une classe est globale et accessible partout dans le script qui l'a créée.
- **N.B :** toute modification d'une variable locale opérée dans le corps d'une fonction n'a aucun effet sur une variable externe à la fonction et portant le même nom.
- Pour utiliser la valeur d'une variable globale dans une fonction, il faut la déclarer dans le corps de la fonction avec le mot-clé **global**.
- **Exemple :**

Voir Tableau 29

Fonctions

Portée des variables

Les variables statiques :

- Pour conserver la valeur précédemment affectée à une variable locale entre deux appels d'une même fonction, il faut déclarer la variable comme statique avec le mot-clé **static**, et ce avant de l'utiliser dans le corps de la fonction.
- L'utilisation typique des variables statiques concerne les fonctions qui effectuent des opérations de cumul.
- **N.B :** Une variable déclarée comme statique ne conserve toutefois sa valeur que pendant la durée du script (i.e. exécution d'une page).
- **Exemple :**

Voir Tableau 30

Fonctions

Passage de paramètres par référence

- Lorsque les paramètres sont passés par valeur, une copie des variables est utilisée par la fonction. Ainsi, les modifications apportées aux valeurs des paramètres passés ne sont pas visibles à l'extérieur de la fonction.
- Grâce au passage des paramètres par référence, les modifications effectuées dans le corps d'une fonction sont répercutées à l'extérieur.
- Le passage de paramètres par référence peut être :
  - Systématique ;
  - occasionnel.
- **Exemple :**

Voir Tableau 31

Fonctions

Fonctions dynamiques

- PHP offre la possibilité de travailler avec des noms de fonctions dynamiques qui peuvent être variables et donc dépendants de l'utilisateur du site ou de l'interrogation d'une base de données.
- **Exemple :**

Voir Tableau 32

Fonctions

Fonctions conditionnelles

- Une fonction est conditionnelle est une fonction définie à l'intérieur d'un bloc **if**.
- Une fonction conditionnelle n'est utilisable que si l'instruction **if** qui la contient a été exécutée et l'expression conditionnelle contenue dans **if** a la valeur booléenne **TRUE**.
- **Exemple :**

Voir Tableau 33

Fonctions

Fonctions récursives

- Une fonction est dite récursive si, à l'intérieur de son corps, elle s'appelle elle-même avec une valeur de paramètre différent (sinon elle boucle).
- Chaque appel constitue un niveau de récursivité.
- **Exemple :** la fonction qui retourne la factorielle d'un nombre entier n.

Voir Tableau 34

Plan du chapitre 5

- Fonctions mathématiques
- Opérateurs booléens
- Instructions conditionnelles
- Instructions de boucle
- Gestion des erreurs
- Chaînes de caractères
- Tableaux
- Formulaires
- Fonctions
- **Dates**

Dates

Définition d'une date

int time() ;
retourne le timestamp de l'instant présent en secondes par rapport à une date d'origine arbitraire, correspondant au 1er janvier 1970 00 h 00 m 00 s.

int mktime(int heure, int minute, int seconde, int mois, int jour, int année, int été)
retourne le timestamp correspondant à la date définie par les valeurs entières passées en paramètres. Le dernier paramètre doit valoir 1 pour l'heure d'hiver, 0 pour l'heure d'été et – 1, valeur par défaut, si vous ne savez pas.

Exemple :

Voir Tableau 35

Dates

Vérification d'une date

boolean checkdate(int mois, int jour, int année) ;
retourne une valeur booléenne TRUE si la date existe et FALSE dans le cas contraire.

- Exemple :

Voir Tableau 36

Dates

Affichage d'une date

string date(string format_de_date,[int timestamp]) ;

retourne une chaîne contenant des informations de date dont la mise en forme est définie par des caractères spéciaux. La date retournée correspond à celle du timestamp passé en deuxième paramètre ou, si ce dernier est omis, à celle de l'instant en cours. Pour afficher un des caractères spéciaux indépendamment de sa fonction de formatage, il faut le faire précéder d'un antislash. Exemple, \h affiche le caractère « h » et non le nombre d'heure. Pour afficher les caractères « n » et « t », il faut écrire \\n et \\t car \n et \t sont employés pour le saut de ligne et la tabulation.

☐ **Exemple :**

```
echo "Aujourd'hui ",date("l, d F Y \i\l \e\s\\t  H:i:s ");
$numjour = date("w");
echo $numjour ;
```

```
Aujourd'hui Monday, 20 October 2008 il est 23:36:27
1
```

Dates

Affichage d'une date

- Caractères de définition du format d'affichage :

Voir Tableau 37

array getdate([int timestamp]) ;
retourne un tableau contenant toutes les informations de date. Si le paramètre timestamp est omis, la fonction getdate() retourne les informations sur la date en cours.

- Exemple :

```
$jour = getdate();
echo "Aujourd'hui {$jour["weekday"]} {$jour["mday"]}
{$jour["month"]} {$jour["year"]}";
```

```
Aujourd'hui Monday 20 October 2008
```

- Clés du tableau retourné par la fonction getdate() :

Voir Tableau 38

Plan du chapitre 5

- Fichiers
- Sessions
- L'envoi des e-mails
- Accès à une base MySQL avec PHP

Fichiers

- Les informations en provenance des visiteurs d'un site sont récupérées dans des variables créées côté serveur dont les valeurs sont perdues dès que l'exécution de script est terminée.
- Deux moyens ont été envisagés pour la sauvegarde des données récupérées durant une connexion afin de pouvoir les réutiliser plus tard, à savoir :
  - **Les fichiers :** type de stockage utilisé pour des quantités de données de taille modeste et quand il n'est pas nécessaire d'effectuer par la suite des recherches complexes parmi elles.
  - **Les bases de données :** permet de stocker des quantités importantes de données et d'effectuer des recherches pointues.

Fichiers

Création d'un fichier

boolean touch(string "nom_fichier"[,integer timestamp]) ;

Crée un fichier vide et lui affecte la date passée en deuxième argument sous la forme d'un timestamp UNIX comme date de dernière modification.

resource tmpfile() ;

crée un fichier temporaire sur le serveur utilisé pour stocker des informations qui ne seront conservées que pendant la durée de la session ouverte par un client ou jusqu'à ce que le fichier soit explicitement fermé au moyen de la fonction **fclose()**.

Exemple :

```
if(!file_exists("monfich.txt"))   // file_exists() permet de vérifier l'existence d'un fichier
{
   touch("monfich.txt",time());
}
```

Fichiers

Ouverture d'un fichier

resource fopen(string $nom, string mode) ;
Permet d'ouvrir, selon le mode indiqué, le fichier dont le nom est passé en paramètre dans la variable $nom. Elle retourne un identifiant de type resource dont la forme est la suivante : id#n, n étant un entier incrémenté de 1 à chaque ouverture de fichier par le même script (la première valeur est toujours 1). En cas d'échec la fonction retourne FALSE.

Valeurs possibles du paramètre mode :	
"r"	le fichier est ouvert en lecture seule, et la lecture commence au début du fichier.
"r+"	le fichier est ouvert en lecture et en écriture, et ces opérations commencent au début du fichier.
"w"	le fichier est ouvert en écriture seule, et l'écriture commence au début du fichier. si le fichier n'existe pas il est créé automatiquement. S'il existe, son contenu antérieur est effacé.

Fichiers

Ouverture d'un fichier

Valeurs possibles du paramètre mode :	
"w+"	Le fichier est ouvert en lecture et en écriture, et ces opérations commencent au début du fichier. si le fichier n'existe pas il est créé automatiquement. S'il existe, son contenu antérieur est effacé.
"a"	le fichier est ouvert en écriture seule, et les données sont écrites en fin de fichier, à la suite de celles qui existent déjà ou au début s'il est vide. Si le fichier n'existe pas, il est créé.
"a+"	le fichier est ouvert en lecture et en écriture, et les données sont écrites en fin de fichier, à la suite de celles qui existent déjà. La lecture s'effectue à partir du début du fichier. Si le fichier n'existe pas, il est créé.

Exemple :

```
$id_file = fopen("monfichier.txt","a");
if(!$id_file) echo "Erreur d'accès au fichier";
```

Fichiers

Fermeture d'un fichier

boolean fclose($id_file)

Permet de fermer un fichier ouvert.

Fichiers

Verrouillage de fichier

- Si plusieurs utilisateurs accèdent au même fichier à partir du même script ou de deux scripts différents et y effectuent simultanément des opérations de lecture ou d'écriture, le fichier risque de devenir inutilisable pour chacun d'eux.
- Afin d'éviter la corruption des fichiers, il faut que les scripts qui y accèdent définissent une priorité d'accès au premier script effectuant une opération sur le fichier ce qui empêche les autres d'y accéder et le modifier tant que le fichier n'est pas fermé.
- Il ainsi possible de verrouiller le fichier en bloquant partiellement ou complètement l'accès pour d'autres utilisateurs pendant qu'un script y accède, jusqu'à ce qu'il soit libéré pour d'autres accès concurrents.
- Certes le verrouillage des fichiers présente l'avantage de la sécurité, mais il a aussi l'inconvénient d'interdire les accès simultanés ce qui ralentit l'accès aux fichiers pour le stockage.

Fichiers

Verrouillage de fichier

- Ainsi, l'utilisation des fichiers est limitée aux sites qui ont un trafic limité ou aux opérations de maintenance ou de sauvegarde.
- La fonction utilisée pour le verrouillage d'un fichier est :

boolean flock(resource $id_file, int N) ;

Le premier paramètre est l'identificateur de fichier retourné par la fonction fopen(). Le second est une constante entière nommée qui définit le mode de verrouillage du fichier :

- **LOCK_SH :** bloque l'écriture dans le fichier mais laisse le libre accès en lecture à tous les utilisateurs. La constante LOCK_SH a la valeur 1.
- **LOCK_EX :** bloque l'écriture et la lecture du fichier par un autre script. Le script en cours a donc l'exclusivité. Une tentative d'accès simultané retourne la valeur FALSE. La constante LOCK_EX a la valeur 2.
- **LOCK_UN :** libère le verrou installé précédemment. Vous ne devez surtout pas oublier d'effectuer cette action après les opérations réalisées sur le fichier, faute de quoi le blocage subsiste. La constante LOCK_UN a la valeur 3.

Fichiers

Verrouillage de fichier

Exemple :

```
$id_file = fopen("monfichier.txt","mode");
flock($id_file,LOCK_SH ou LOCK_EX);
//ou encore
//flock($id_file,1 ou 2);
//opérations de lecture et/ou d'écriture
flock($id_file(LOCK_UN);
//ou encore
//flock($id_file,3);
fclose($id_file);
```

Fichiers

Écriture dans un fichier

```
integer fwrite(resource $id_file,string "chaine" [,int N]) ;
integer fputs(resource $id_file, string "chaine" [,int N]) ;
```

Elles écrivent toutes deux le texte contenu dans "chaine" dans le fichier identifié par la variable $id_file. Lorsque le paramètre N est précisé, seuls les N premiers caractères de la chaîne sont écrits dans le fichier.

Exemple 1 :

Voir Tableau 46-1

Exemple 2 :

Voir Tableau 46-2

Fichiers

Lecture de fichiers

string fgets(resource $id_file,integer nombre_octets) ;
lit le fichier depuis son début et retourne une chaîne de caractères d'une longueur maximale égale au paramètre nombre_octets. La lecture s'arrête quand ce nombre d'octets lu est atteint ou avant qu'il soit atteint, si le caractère "\n" est rencontré dans le fichier.
string fread(resource $id_file,integer nb_caracteres) ;
lit le fichier depuis son début et retourne à chaque appel une chaîne de caractères contenant exactement le nombre de caractères précisé dans le second paramètre, sauf si la fin du fichier est atteinte ou si le caractère "\n" est rencontré. Son utilisation est adaptée à des fichiers dans lesquels vous avez préalablement enregistré des données de longueur fixe par paquets égaux.
string fgetc(resource $id_file) ;
lit un caractère à la fois dans le fichier texte.

Fichiers

Lecture de fichiers

array fgetcsv(resource $id_file, integer nombre_octets, string "séparateur") ;
lit dans le fichier identifié par $id_file au maximum le nombre de caractères précisé à l'aide du deuxième paramètre. Elle retourne directement un tableau de chaînes de caractères et non une seule chaîne à chaque appel, comme si vous appliquiez la fonction explode() à une chaîne comprenant un séparateur. La lecture de chaque ligne s'arrête, lors de l'apparition d'un saut de ligne "\n" dans le fichier.
integer fseek(resource $id_file,integer nombre_d'octets) ;
positionne le pointeur de lecture et/ou d'écriture en un point particulier du fichier indiqué par le second paramètre représentant le nombre d'octets par rapport au début, à partir du quel doit commencer la lecture. Elle retourne la valeur booléenne TRUE si l'opération est réussie et la valeur –1 dans le cas contraire.
boolean rewind(resource $id_file) ;
Remet le pointeur au début du fichier.

Fichiers

Lecture de fichiers

integer ftell($id_file) ;
Retourne la position du pointeur en nombre d'octets par rapport au début du fichier.
integer filesize(string "nom_fichier") ;
Retourne la taille totale du fichier en nombre d'octets.
integer readfile(string "nom_fichier") ;
Permet de lire la totalité d'un fichier sans avoir à analyser son contenu. Elle retourne un entier indiquant le nombre total d'octets affichés sur le navigateur. Elle ne nécessite pas l'appel des fonctions fopen() et fclose() d'ouverture et de fermeture.
array file(string "nom_fichier")
Retourne la totalité du contenu du fichier dans un tableau indicé dont chaque élément est constitué d'une seule ligne du fichier. Il suffit d'utiliser une boucle for ou foreach pour afficher chacune des lignes du tableau. Elle ne nécessite pas l'appel des fonctions fopen() et fclose() d'ouverture et de fermeture.
fpassthru($id_file)
permet la lecture d'un fichier dans son intégralité. Elle nécessite l'emploi de fopen() pour ouvrir le fichier mais pas de fclose() pour le fermer.

Fichiers

Lecture de fichiers

Exemple :

Voir Tableau 47

Fichiers

Modifications de fichiers

boolean copy(string "nom_fichier",string "nom_copie") ;

Crée une copie d'un fichier donnée sous un autre nom ou avec une extension différente afin de récupérer l'ensemble des données en cas de problème ayant atteint l'intégrité du fichier. Elle retourne la valeur booléenne TRUE si la copie est réalisée et FALSE en cas de problème d'écriture.

boolean rename(string "nom_actuel",string "nom_futur") ;

Permet de changer le nom d'un fichier existant. Comme avec l'opération Renommer de l'Explorateur Windows, le fichier original n'existe plus sous son nom initial. Elle retourne TRUE si l'opération est effectuée et FALSE dans le cas contraire.

boolean unlink(string "nom_fichier") ;

Supprime définitivement un fichier présent dans le même répertoire que le script qui effectue l'opération. Elle dispose d'un unique paramètre qui est le nom du fichier à supprimer contenu dans une chaîne de caractères. Elle retourne TRUE ou FALSE selon que la suppression est effectuée ou non. Avant de réaliser la suppression, il est impératif de s'assurer que le fichier existe à l'aide de la fonction file_exists(), faute de quoi une erreur d'exécution est produite.

Fichiers

Récupération d'informations sur les fichiers

boolean file_exists(string "nom_fichier") ;
Retourne une valeur booléenne TRUE ou FALSE selon que le fichier existe ou non dans le dossier du script qui l'appelle.
boolean is_file(string nom_fichier) ;
Retourne la valeur booléenne TRUE pour le fichier ouvert identifié par $id_file et FALSE s'il ne s'agit pas d'un fichier.
boolean is_readable(string nom_fichier) ;
Retourne TRUE si le fichier est disponible en lecture et FALSE dans le cas contraire.
boolean is_writable(string nom_fichier)
Retourne TRUE si le fichier est disponible en écriture et FALSE dans le cas contraire.
string filetype(string "nom_fichier")
Retourne la chaîne "file" si le paramètre est un fichier et "dir" si c'est un répertoire.

Fichiers

Récupération d'informations sur les fichiers

integer fileatime(string "nom_fichier") ;
Retourne le timestamp de la date du dernier accès au fichier dont le nom est donné dans la chaîne de caractères passée en paramètre.
integer filemtime(string "nom_fichier") ;
Retourne le timestamp de la dernière modification du fichier dont le nom est donné dans la chaîne de caractères passée en paramètre.
integer filectime(string "nom_fichier") ;
Retourne le timestamp de la dernière modification des permissions du fichier dont le nom est donné dans la chaîne de caractères passée en paramètre.
string realpath(string "nom_fichier") ;
Retourne le chemin d'accès complet à un fichier, en connaissant simplement son nom, dans une chaîne de caractères.
string basename(string "chemin_d'accès") ;
Extrait uniquement le nom d'un fichier en indiquant le chemin d'accès comme paramètre dans la fonction.

Plan du chapitre 5

- Fichiers
- **Sessions**
- L'envoi des e-mails
- Accès à une base MySQL avec PHP

Sessions

Définition

- Il s'agit d'un mécanisme permettant de mettre en relation les différentes requêtes du même client sur une période de temps donnée.
- Les sessions permettent de conserver des informations relatives à un utilisateur lors de son parcours sur un site web. Ainsi, les informations provenant d'une page peuvent être transmises et utilisées dans une autre page.

Sessions

Mécanisme des sessions

L'utilisation du mécanisme des sessions obéit aux étapes générales suivantes :

- Lorsqu'un visiteur arrive sur un site donné, une session lui est créée. Pour chaque session ouverte PHP attribut un nom (**PHPSESSID**) et un identifiant unique (**ID**), qui est une suite de 26 caractères aléatoires (exemple : a02bbffc6198e6e0cc2715047bc3766f). le nom de la session et son identifiant sont contenus dans une constante nommée **SID**. Le **SID** est transmis d'une page à une autre durant la même session de deux manières différentes :
  - soit en étant écrit dans un cookie sur le poste client : il faut pour cela que la directive **session.use_cookies** du fichier **php.ini** ait la valeur 1 et plus fondamentalement que le poste client accepte les cookies.
    N.B : Certains sites de commerce en ligne lance un avertissement du type « pour accéder à ce service, vous devez accepter les cookies ».
  - soit en étant ajouté à l'URL de la page cible d'un lien.

Sessions

Mécanisme des sessions

L'utilisation du mécanisme des sessions obéit aux étapes générales suivantes :

- Une fois la session est créée, une infinité de variables de session peuvent être définies en fonction des besoins et seront accessibles dans toutes les pages du site. Ces variables sont récupérées dans un tableau associatif superglobal **$_SESSION**, dont les clés sont les noms mêmes des variables (exemple : $_SESSION['login'], $_SESSION['passw'] ...). Les données du tableau **$_SESSION** sont stockées dans des fichiers (fichiers texte) ayant pour nom l'identifiant de la session auquel est ajouté le préfixe **sess_** et placés généralement dans le dossier **/tmp** du serveur. Il est à noter que certains hébergeurs obligent les programmeurs de sites Web à créer eux-mêmes sur le serveur un dossier nommé, par exemple, **sessions** pour activer les sessions.

Sessions

Mécanisme des sessions

L'utilisation du mécanisme des sessions obéit aux étapes générales suivantes :

- Lorsque le visiteur se déconnecte du site, la session est fermée après destruction éventuelle de toutes les variables de session. La déconnexion d'un internaute est détectée lorsque :
  - Il ferme son navigateur ;
  - Il visite un autre site Web ;
  - Il clique sur un bouton « Déconnexion » créé par le programmeur du site en cours ;
  - Suite à un timeout : l'internaute est déconnecté automatiquement après un certain moment d'inactivité.

Sessions

Mécanisme des sessions

- La fonction **session_start()** est utilisée pour démarrer le système de sessions. Elle doit être placée au début de chaque page (avant même la balise **<!DOCTYPE>**) afin de d'ouvrir ou de continuer une session.
- La fonction **session_destroy()** est utilisée pour fermer la session en cours. Cette commande supprime toutes les informations relatives à l'utilisateur (i.e. toutes les données enregistrées d'une session).
- La fonction **session_unset()** détruit toutes les variables d'une session.
- Les fonctions **session_name()** et **session_id()** permettent respectivement de récupérer le nom et l'identifiant d'une session ouverte.

Sessions

Exemples d'utilisation des sessions

- Exemple 1 : Pages à accès réservé par une authentification

Voir Tableau 48

- Exemple 2 : Gestion de panier et commande en ligne

Voir Tableau 49

Plan du chapitre 5

- Fichiers
- Sessions
- **L'envoi des e-mails**
- Accès à une base MySQL avec PHP

L'envoi des e-mails

- Il est possible d'envoyer des e-mails du serveur vers le poste client, pour peu que le serveur sur lequel vous hébergez votre site Web vous l'autorise.
- Il est à noter que de nombreux hébergeurs, en particulier les gratuits, désactivent la fonction d'envoi d'e-mail pour éviter le Spam et surtout pour ne pas surcharger leurs serveurs.
- L'envoi des e-mails est possible grâce à la fonction **mail()** dont la syntaxe est la suivante :

boolean mail($dest, $objet, $texte, [$entete]) ; // elle retourne TRUE si le message est expédié et FALSE dans le cas contraire.	
$dest	une chaîne contenant l'adresse e-mail du destinataire. Pour envoyer le même e-mail à plusieurs adresses, il faut séparer chacune d'elles par une virgule.
$objet	une chaîne contenant le texte qui apparaît dans la colonne Objet du logiciel de courrier du destinataire.
$texte	une chaîne donnant le contenu réel du message, qui peut être au format texte ou au format HTML.
$entete	une chaîne contenant les en-têtes nécessaires à l'envoi d'e-mails, lorsque ces derniers ne sont pas au format texte. Chaque en-tête se termine par la séquence "\n" sur un serveur Linux et "\r\n" sous Windows.

L'envoi des e-mails

- Les e-mails envoyés peuvent être au format :
  - Texte ;
  - HTML (exp : mails publicitaires ou d'information envoyés à une liste de personnes inscrites à une liste de distribution (mailing list));
- Comparée à la méthode d'envoi des e-mails au format texte, la méthode d'envoi en HTML présente l'inconvénient d'obliger le destinataire à être en ligne pour pouvoir visualiser correctement les messages envoyés surtout lorsqu'ils contiennent des images ou des sons. Toutefois, cette méthode rend, en contrepartie, les messages moins lourds.

L'envoi des e-mails

- Les en-têtes utilisables dans l'envoi des mails au format texte ou HTML sont :

En tête	Définition
From:	Adresse de l'expéditeur de l'e-mail si vous souhaitez qu'elle soit différente de celle qui est écrite dans le corps du message.
cc:	Adresse e-mail du destinataire en copie. S'il y en a plusieurs, elles doivent être séparées par des virgules.
bcc:	Adresse du destinataire en copie cachée
Reply-To:	Adresse à laquelle parviendra la réponse éventuelle du destinataire s'il rédige son e-mail en utilisant le bouton Répondre.
X-Mailer:	Nom du logiciel d'envoi du courrier
Date:	Date de l'e-mail au format JJ MM AAAA h:m:s +0N00, dans laquelle N est le décalage horaire.

L'envoi des e-mails

- Le quatrième paramètre de la fonction **mail()** est optionnel lorsque l'e-mail est envoyé au format texte et devient obligatoire dans le cas d'un envoi au format HTML incorporant divers types de données (images, sons, texte HTML...).
- Dans les e-mails envoyés au format HTML, on ajoute au quatrième paramètre de la fonction **mail()** les en-têtes **MIME** séparés par la séquence "\n".

En tête	Définition
`MIME-Version:`	Indique que le contenu de l'e-mail est conforme aux spécifications MIME ainsi que la version utilisée (actuellement 1.0 ou 1.1). Cet en-tête doit être écrit le premier : `MIME-Version: 1.0`
`Content-Type:`	Définit le type de contenu de l'e-mail à l'aide d'un type MIME ainsi que le jeu de caractères à utiliser : `Content-Type: text/html;charset=iso-8859-1`
`Content-Transfer-Encoding:`	Définit le mode de codage des documents, en particulier des images liées à l'e-mail. Il y a plusieurs valeurs possibles, mais il est préférable de choisir la valeur `8bit` : `Content-Transfer-Encoding: 8bit`

L'envoi des e-mails

- Exemple 1 : Envoi d'un e-mail de confirmation au format texte

Voir Tableau 50

- Exemple 2 : Envoi d'e-mail d'annonce d'un grand événement au format HTML

Voir Tableau 51