Programmation Web

Enseignant: Achouri Khalil

Email: achourikhalil@gmail.com

MCours.com

Chapitre 4

PHP

Documentation en ligne

Pour obtenir en ligne toute la documentation officielle (en français) sur une commande, tapez l'URL suivante dans la barre d'adresse de votre navigateur Internet :

http://fr.php.net/

Et rajouter en fin d'URL le nom de la commande.

Exemple :

http://fr.php.net/echo

Historique

Il a été créé en 1994 par Rasmus Lerdorf pour les besoins des pages web personnelles (livre d'or, compteurs, etc.). A l'époque, PHP signifiait Personnal Home Page.

C'est un langage incrusté au HTML et interprété (PHP3) ou compilé (PHP4 et PHP5) côté serveur. Il dérive du C et du Perl dont il reprend la syntaxe. Il est extensible grâce à de nombreux modules et son code source est ouvert. Comme il supporte tous les standards du web et qu'il est gratuit, il s'est rapidement répandu sur la toile.

En 1997, PHP devient un projet collectif et son interpréteur est réécrit par Zeev Suraski et Andi Gutmans pour donner la version 3 qui s'appelle désormais PHP : Hypertext Preprocessor (acronyme récursif à l'exemple du système Open Source Linux : Is Not UniX).

Il existe par ailleurs des applications web prêtes à l'emploi (PHPNuke, PHP SPIP, PHPSlash…) permettant de monter facilement et gratuitement son portail. En juillet 2000 plus de 300.000 sites tournaient déjà sous PHP !

Intégration d'un script PHP dans une page HTML

Les pages web sont au format html (extension .htm ou .html). Les pages web dynamiques générées avec PHP sont au format php (extension .php). Le code source PHP est directement inséré dans le fichier html grâce à la balise (ou TAG):

<?php Liste d'insctructions ?>

Exemple:

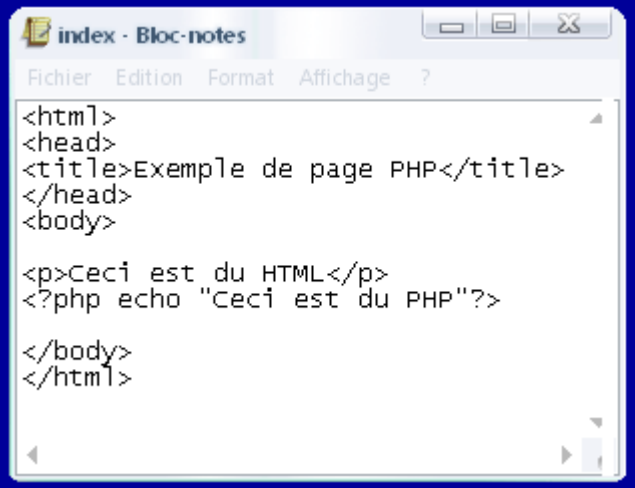
```
<html><body>
<?php  echo ''Bonjour'';  ?>
</body></html>
```

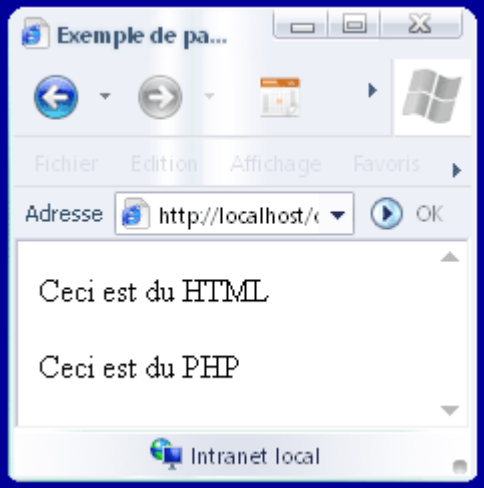
Autres balises:

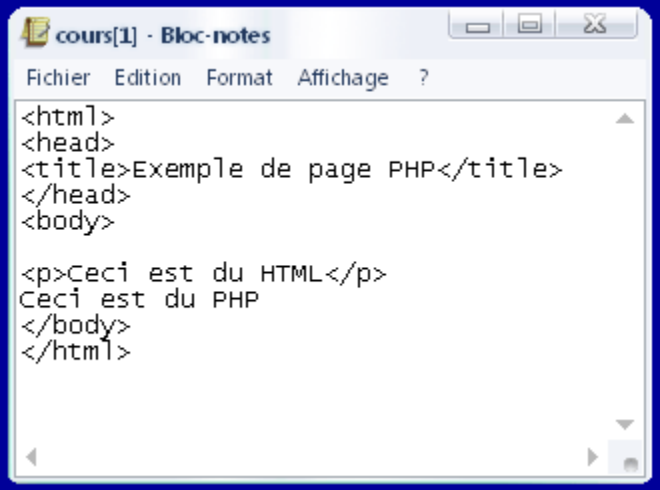
<? … ?>	(balise simple)
<script language=''PHP''> … </script>	(balise de script)
<% … %>	(balise ASP)

Exemple de scripts

Soit la page nommée index.php:







Code source (coté serveur)

Aperçu sur le navigateur (coté client)

Code source (coté client)

On remarque que les balises PHP qui existent sur la page d'origine (coté serveur) ont disparu au coté client => confidentialité du code PHP

Commentaires PHP

Un script php se commente comme en C :

Exemple :

```
<?php

// commentaire de fin de ligne

/* commentaire
sur plusieurs
lignes */

?>
```

Tout ce qui se trouve dans un commentaire est ignoré. Il est conseillé de commenter largement ses scripts.

Variables, types et opérateurs

Le typage des variables est implicite en PHP. Il n'est donc pas nécessaire de déclarer leur type au préalable ni même de les initialiser avant leur utilisation.

Les identificateurs de variable sont précédés du symbole « $ » (dollars). Exemple : $toto.

Les variables peuvent être de type entier (integer), réel (double), chaîne de caractères (string), tableau (array), booléen (boolean).

Il est possible de convertir une variable en un type primitif grâce au cast(1) (comme en C).

Exemple :

```
$str = ”12”;              // $str vaut la chaîne ‘’12”
$nbr = (int)$str;                  // $nbr vaut le nombre 12
```

(1) : Le cast est une conversion de type. L'action de caster consiste en convertir une variable d'un type à un autre.

Les constantes

Les constantes PHP ne sont préfixées par aucun symboles. Elles sont déclarée de la façon suivante:

```
define(“id_constante”,“valeur”);
//  ou bien define(id_constante, “valeur”);
```

Exemple:

```
define(“nom”, “ENSA”);
```

Dans ce cas la constante nom vaut la valeur ENSA

Les constantes ne sont définie qu’une seule fois dans le script. Leurs valeurs restent inchangées et leurs identificateurs ne sont pas sensibles à la casse (contrairement aux variables).

Les opérateur I

- Opérateurs arithmétiques :

+ (addition), **-** (soustraction), ***** (multiplié), **/** (divisé), **%** (modulo), **++** (incrément), **--**(décrément).

- Opérateurs d'assignement :

= (affectation), *= ($x*=$y équivalent à $x=$x*$y), /=, +=, -=, %=

- Opérateurs logiques :

and, && (et), or, || (ou), xor (ou exclusif), ! (non)

- Opérateurs de comparaison :

== (égalité), **<** (inférieur strict), **<=** (inférieur large), **>**, **>=**, **!=** (différence)

Les opérateur II

Il existe un opérateur très spécial qui équivaut à une structure conditionnelle complexe if then else à la différence qu'il renvoie un résultat de valeur pouvant ne pas être un booléen : l'opérateur ternaire.

Syntaxe :

(condition)?(expression1):(expression2);

Si la condition est vrai alors évalue et renvoie l'expression1 sinon évalue et renvoie l'expression2.

Exemple :

$nbr = ($toto>10)?($toto):($toto%10);

Dans cet exemple, la variable $nbr prend $toto pour valeur si $toto est strictement supérieur à 10, sinon vaut le reste de la division entière de $toto par 10.

Les références

On peut à la manière des pointeurs en C faire référence à une variable grâce à l'opérateur & (ET commercial).

Exemple :

```
$var1 = 100;       //  la variable $var1 est initialisée à la valeur 100

$var2 = &$var1;  //  la variable $var2 fait référence à $var1

$var1++;           //  on change la valeur de $var1

echo $var2;        //  qui est répercutée sur $var2 qui
                   //  vaut alors 101
```

Fonctions agissant sur les variables I

Quelques fonctions :

empty($var) : teste si la variable est vide ou non (renvoie true ou false)

isset($var) : teste si la variable existe ou non (renvoie true ou false)

unset($var) : détruit une variable

gettype($var) : retourne le type de la variable

settype($var,''type'') : convertit la variable d'un type à un autre (CAST)

is_int() : teste si la variable est de type entier ou non (renvoie true ou false)

is_long(), is_double(), is_string(), is_array(), is_bool(), is_float(), is_numeric(), is_integer() : Idem que is_int().

Fonctions agissant sur les variables II

Une variable peut prendre comme identificateur la valeur d'une autre variable en utilisant la syntaxe suivante:

${$var} = valeur

Exemple:

```
$variable = "bonjour";
${$variable} = 2007;
echo $bonjour;  // renvoie 2007
```

Dans cet exemple l'identificateur de le deuxième variable a pris la valeur de la première variable "bonjour".

Mathématiques I

PHP dispose d'une grande série de fonctions mathématiques.

Quelques fonctions:

- abs($x) : renvoie la valeur absolue de $x
- ceil($x) : arrondi supérieur
- floor($x) : arrondi inférieur
- pow($x,$y) : x exposant y
- round($x,$i) : arrondi de x à la i^{ème} décimale
- max($a, $b, $c …) : retourne l'argument de valeur maximum
- min($a, $b, $c …) : retourne l'argument de valeur minimum
- pi() : retourne la valeur de PI

Et aussi : cos, sin, tan, exp, log, sqrt… ainsi que des constantes tels que M_PI, M_E

Mathématiques II

Nombres aléatoires

La fonction rand($x,$y) permet de générer un nombre aléatoire compris entre $x et $y s'ils sont déclarés sinon génère un nombre aléatoire compris entre 0 et RAND_MAX

srand() initialise le générateur aléatoire

getrandmax() retourne la valeur du plus grand entier pouvant être généré.

La fonction rand() se base sur de vieux algorithmes. elle a donc un comportement qui peut être prévisible. La nouvelle fonction mt_rand() est basée sur la cryptographie. Elle est plus rapide et, pratiquement, imprévisible.

Les fonctions qui dérivent de la même famille sont mt_srand() et mt_getrandmax().

MCours.com

Mathématiques III

Formatage d'un nombre

number_format ($nbr,$dec,$a,$b) : retourne une chaîne de caractères représentant le nombre $nbr avec $dec décimales après formatage. La chaîne $a représente le symbole faisant office de virgule et $b le séparateur de milliers.

Par défaut, le formatage est anglophone : $a = '.' et $b = ','

Cette fonction est très utile pour représenter les nombres élevés au format francophone

Exemples:

```
number_format (1000000.3333);              // affiche 1,000,000
number_format (1000000.3333,2);            // affiche 1,000,000.33
number_format (1000000.3333,2,",","."); // affiche 1.000.000,33
```

Chaînes de caractère

Une variable chaîne de caractères n'est pas limitée en nombre de caractères. Elle est toujours délimitée par des simples quotes (') ou des doubles quotes (").

Exemples :

$etablissement = ''ENSA'';
$ville = 'Marrakech';

Les doubles quotes permettent l'évaluation des variables et caractères spéciaux contenus dans la chaîne (comme en C) alors que les simples ne le permettent pas.

Exemples :

```
echo ''$etablissement'';        // affiche ENSA
echo '$etablissement';          // affiche $etablissement
```

Chaînes de caractère

Concaténation de plusieurs chaînes

Exemple 1 :

```
$var1 = ‘’Hello’’;
$var2 = ‘’Word’’;
echo $var1.$var2;    // imprime HelloWord
```

Exemple 2 :

```
$etab = ‘’ENSA’’;
echo $etab.‘’Marrakech’’;   // Affiche ENSAMarrakech
```

Exemple 3 :

```
$var = ‘Université’;
$var.= ‘’ Cadi Ayyad’’
echo $var;      //  affiche Université Cadi Ayyad
```

Chaînes de caractère

Affichage

echo est la fonction d'affichage la plus utilisée en PHP.

Exemple:

```
echo "Bonjour"   // Affiche Bonjour
```

Cependant ils existent d'autres fonction plus ou moins utilisées. À savoir: print() et printf()

```
print(" chaîne");     //   équivalent à echo "chaîne"
Printf("chaîne","format")  // affichage formaté comme en C
```

Exemple:

```
Printf("ENSA %s","Marrakech");  // imprime ENSA Marrakech
```

Chaînes de caractère

Affichage – suite –

On peut afficher des chaînes de caractères s'étalant sur plusieurs lignes en utilisant la syntaxe suivante:

```
$str = <<<EOD
Cette syntaxe
permet de déclarer une chaîne
de caractères qui occupe
plusieurs lignes
EOD;

echo $str; // Cette syntaxe permet de déclarer une
           // chaîne de caractères qui occupe plusieurs lignes
```

Chaînes de caractère

Quelques fonctions

- strlen($str) : retourne la taille de la chaîne $str en caractères.
- strtolower($str) : conversion en minuscules
- strtoupper($str) : conversion en majuscules
- ucfirst($str) : convertit la première lettre en majuscule
- trim($str) : suppression des espaces de début et de fin de chaîne
- rtrim($str), chop($str): suppression des espaces de fin de la chaîne.
- substr($str,$i,$j) : retourne une sous chaîne de $str de taille $j et débutant à la position $i
- stristr($str,$char): retourne une sous chaîne de $str commençant de la position $char et allant jusqu'à la fin de la chaine.
- addslashes($str) : déspécialise les caractères spéciaux
- ord($char) : retourne la valeur ASCII du caractère $char
- chr($int): retourne le caractère dont le code ASCII est $int
- bin2hex($str): retourne la représentation hexadécimale de chaque lettre de $str.

Les structures de contrôle

La syntaxe des structures de contrôle est la même que celle en langage C.

Structures conditionnelles

```
if( condition1 )
    {
        traitement 1
    }
elseif(condition 2-1)
        {
            traitement 2-1
        }
    else
        {
            traitement 2-2
        }
```

Les structures de contrôle

Structures conditionnelles – suite –

```
switch( paramètre_index )
{
   case valeur 1 : { traitement 1 } break;
   case valeur 2 : { traitement 2 } break;
   ...
   default : { traitement par défaut }
}
```

Structures de boucles

```
for( $i=$n ; $i<$m ; $i++ )
{
   traitement
}
```

Les structures de contrôle

Structures de boucles

```
while( condition )
{
   traitement
}

do
{
   traitement
}
while( condition )
```

Les structures de contrôle

L'instruction break permet de quitter prématurément une boucle.

Exemple :

```
while($nbr < 10) {
    echo $nbr."<br>";
    if($nbr == 5)
        break;
    $nbr++;
}
```

L'instruction continue permet d'ignorer le traitement associé à une valeur de la boucle

Exemple :

```
for($i=1; $i<=10; $i++) {
    if($i==5)
        continue;
    echo $i;
}
```

Les tableaux

La fonction array() permet de créer des tableaux dynamiques qui seront exploités par les scripts PHP. Un tableau array est temporaire et ne reste généré que le temps de déroulement du script.

Les tableaux array servent souvent à stoker les données provenant d'une base de données en attendant le traitement. Elles permettent aussi aux fonctions de retourner plusieurs résultats au lieu d'une seule.

Remarque:

les tableaux dynamiques ou arrays n'ont rien à voir avec les tableaux HTML, qui servent à la mise en forme des données de la page Web.

Les tableaux

Pour initialiser un tableau on utilise plusieurs méthodes:

Méthode 1: (classique)

$tableau= array($val1,$val2,$val3,…);

Exemple:

$tab=array(‘’ENSA’’,2007,’Marrakech’);

dans ce cas $tab[0]=‘’ENSA’’

$tab[1]=2007

$tab[2]=‘Marrakech’

Méthode 2: (initialisation directe)

$tableau[0]=$val0;

$tableau[1]=$val1;

$tableau[10]=$val10; …

Les tableaux

Méthode 3: (initialisation directe implicite)

$tableau[]=$val0; (sous-entend $tableau[0]=$val0)

$tableau[]=$val1; (sous-entend $tableau[1]=$val1)

$tableau[]=$val2; …

L'appel d'un élément du tableau se fait à l'aide de son indice

Exemple:

Echo $tab[0];

Parcours d'un tableau

```
$i=0;
while($i <= count($tab)) // count() retourne le nombre d'éléments
{
  echo $tab[$i].”<br>”;
  $i++;
}
```

Les tableaux

Parcours d'un tableau

La méthode la plus simple pour parcourir un tableau consiste à l'utilisation de la boucle foreach

La syntaxe est la suivante:

```
foreach($tableau as $element)
{
  traitement;
}
```

La variable $element prend pour valeurs successives tous les éléments du tableau nommé $tableau.

Exemple:

```
foreach($tab as $elem)
{
  echo $elem.''<br>'';
}
```

Les tableaux

Fonctions manipulant les tableaux

count(), sizeof() : retournent le nombre d'éléments du tableau

in_array($var,$tab) : dit si la valeur de $var existe dans le tableau $tab

list($var1,$var2…) : transforme un tableau en liste de variables.

range($i,$j) : retourne un tableau contenant un intervalle de valeurs

shuffle($tab) : mélange les éléments d'un tableau

sort($tab) : trie alphanumérique les éléments du tableau

rsort($tab) : trie alphanumérique inverse les éléments du tableau

implode($str,$tab), join : retournent une chaîne de caractères contenant les éléments du tableau $tab joints par la chaîne de jointure $str

explode($delim,$str) : retourne un tableau dont les éléments proviennent de la chaîne $str et dont le délimiteur est $delim

array_merge($tab1,$tab2,$tab3…) : concatène les tableaux passés en arguments

array_rand($tab) : retourne un élément du tableau au hasard

Les tableaux

Dans un tableau associatif on associe à chacun de ses éléments une clé dont la valeur est de type chaîne de caractères.

L'initialisation d'un tableau associatif est similaire à celle d'un tableau normal.

Exemple 1 :

```
$personne = array(''Nom'' => ''César'', ''Prénom'' => ''Jules'');
```

Exemple 2 :

```
$personne[''Nom''] = ''César'';
$personne[''Prénom''] = ''Jules'';
```

Ici à la clé ''Nom'' est associée la valeur ''César''.

Les tableaux

Parcours d'un tableau associatif

$personne = array(''Nom'' => ''César'', ''Prénom'' => ''Jules'');

Exemple 1 :

```
foreach($personne as $elem) {
  echo $elem;
}
```

Ici on accède directement aux éléments du tableau comme c'est le cas pour les tableaux indexés

Exemple 2 :

```
foreach($personne as $key => $elem) {
  echo ''$key : $elem'';
}
```

Ici on accède simultanément aux clés et aux éléments.

Les tableaux

Quelques fonctions

array_count_values($tab) : retourne un tableau associatif contenant les valeurs du tableau $tab comme clés et leurs fréquence comme valeur (utile pour évaluer les redondances)

array_keys($tab) : retourne un tableau contenant les clés du tableau associatif $tab

array_values($tab) : retourne un tableau contenant les valeurs du tableau associatif $tab

array_search($val,$tab) : retourne la clé associée à la valeur $val

L'élément d'un tableau peut être un autre tableau.

Les tableaux associatifs permettent de préserver une structure de données.

Les tableaux

Quelques fonctions

reset($tab) : place le pointeur sur le premier élément

current($tab) : retourne la valeur de l'élément courant

next($tab) : place le pointeur sur l'élément suivant

prev($tab) : place le pointeur sur l'élément précédant

each($tab) : retourne la paire clé/valeur courante et avance le pointeur

Exemple :

```
$colors = array(''red'', ''green'', ''blue'');
$nbr = count($colors);
reset($colors);
for($i=1; $i<=$nbr; $i++) {
    echo current($colors).''<br>'';
    next($colors);
}
```

Les fonctions I

Une fonction est un sous-programme qui permet d'effectuer un ensemble d'instructions par simple appel dans le corps du programme principal.

Comme tout langage de programmation. PHP permet l'écriture des fonction.

La syntaxe est la suivante:

```
function ma_fonction($argument1, $argument2,…)
{
Liste des instructions;
Return ($valeur_de_retour);
}
```

Une fonction peut ne pas avoir de valeurs d'entrée (arguments) ou encore retourner de valeurs (le cas d'une fonction d'affichage de message d'erreur).

Les fonctions II

Variable locale et variable globale d'une fonction

Toute variable déclarée au sein d'une fonction est de portée locale de cette fonction. Elle n'est pas reconnue dans le reste du programme.

global permet de travailler sur une variable de portée globale au programme. Le tableau associatif $GLOBALS permet d'accéder aux variables globales du script ($GLOBALS[''var''] accède à la variable $var).

Exemple :

```
function change() {
    global $var;        // définit $var comme globale
    $GLOBALS[''toto''] ++; // incrémente la variable globale $toto
    $var++;  // cela sera répercuté dans le reste du programme
}
```

Les fonctions III

On peut donner une valeur par défaut aux arguments lors de la déclaration de la fonction.

Exemple :

```
function Set_Color($color=‘’black’’) {
    global $car;
    $car[‘’color’’] = $color;
}
```

Forcer le passage de paramètres par référence

Exemple :

```
function change(&$var) { // force le passage par référence
    $var += 100;            // incrémentation de +100
}
$toto = 12;       // $toto vaut 12
change($toto);
echo $toto;       // $toto vaut 112
```

Les fonctions IV

Même sans paramètre, un entête de fonction doit porter des parenthèses ( ). Les différents arguments sont séparés par une virgule et le corps de la fonction est délimité par des accolades { }.

Quelques exemples :

```
function afficher($str1, $str2) {  // passage de deux paramètres
    echo ‘’$str1, $str2’’;
}

function bonjour() {               // passage d’aucun paramètre
    echo ‘’Bonjour’’;
}

function GetColor() {              // retour d’une variable
return ‘’black’’;
}
```

Les fonctions V

En version 3 du PHP (version interprétée), une fonction ne peut être appelée qu'après sa définition. En version 4 et 5 (compilées) une fonction peut être appelée avant sa définition.

Exemple :

```
function f1() {              // définition de la fonction f1
    echo ‘’fonction f1…’’;
}
f1();                  // appel de la fonction f1 définie plus haut
f2();                  // appel de la fonction f2 pas encore définie
function f2() {              // définition de la fonction f2
    echo ‘’fonction f2’’;
}
```

Cet exemple affichera : fonction f1…fonction f2!.

Les fonctions VI

Il est possible de faire retourner plusieurs valeurs d'une fonction en utilisant les tableaux ayant pour éléments les variables à retourner. Dans l'appel de la fonction, il faudra alors utiliser la procédure list() qui prend en paramètre les variables à qui ont doit affecter les valeurs retournées. On affecte à list() le retour de la fonction.

Exemple :

```
function trigo($nbr) {
    return array(sin($nbr), cos($nbr), tan($nbr));  // retour d'un
                                                    // tableau
}
$r = M_PI;
list($a, $b, $c) = trigo($r);      /* affectation aux variables $a,$b et
          $c des éléments du tableau retourné par la fonction trigo */
```

Les inclusions

On peut inclure dans un script PHP le contenu d'un autre fichier.

require insert dans le code le contenu du fichier spécifié même si ce n'est pas du code php. (équivalent au préprocesseur *#include* du C)

Syntaxe:

require(''fichier.php'');

include évalue et insert à chaque appel (même dans une boucle) le contenu du fichier passé en argument.

Syntaxe:

include(''fichier.php'');

Les inclusions sont très utiles lors de l'appel d'une base de donnée. Les paramètres de connexions sont stockés sur un fichiers à part.

Arrêt prématuré

Pour stopper prématurément un script, il existe deux fonctions.

die arrête un script et affiche un message d'erreur dans le navigateur.

Exemple :

```
if(mysql_query($requette) == false)
    die(''Erreur de base de données à la requête'');
```

exit l'arrête aussi mais sans afficher de message d'erreur.

Exemple :

```
function mafonc() {
    exit();
}
```

MCours.com

Ces fonctions stoppent tout le script et pas seulement le bloc en cours.

Les sessions I

Les **sessions** permettent le passage des variables (variables de session) d'une page à une autre tout au cours de la visite du site web.

Les variables de session ne sont pas visibles dans l'URL et peuvent être de types différents (tableau, float…).

Cette méthode permet de conserver les paramètres de connexions d'un visiteur et conserver ses choix (sauvegarder son panier e-commerce par exemple)

Les informations de sessions sont conservées en local sur le serveur tandis qu'un identifiant de session est posté sous la forme d'un cookie chez le client.

Les sessions II

Fonctions

session_start() : démarre une session

session_destroy() : détruit les données de session et ferme la session

session_start() doit être présente sur toutes les pages qui utilisent les variables de sessions et doit être déclarée avant tout envoie au navigateur du client.

Une variable de session est manipulée de la façon suivante:

$_SESSION[”nom_de_la_variable”]

Exemple:

$_SESSION[”etablissement”]=”ENSA”;

Les variables d'environnement I

Les variables d'environnement sont, comme leur nom l'indique, des données permettant au programme d'avoir des informations sur son environnement. L'environnement dans le cas du script PHP est :

- Le client
- Le serveur

Les variables d'environnement permettent d'avoir des informations sur le type de serveur, l'emplacement de la page en cours (l'arborescence sur le serveur), la date à laquelle le script a été appelé, l'adresse IP du client, nom du hôte client, le type de navigateur du client, la page qui a renvoyé le navigateur au script en cours…

L'exécution de la fonction phpinfo() permet de lister les variables d'environnement supportés par le serveur

Les variables d'environnement II

Quelques variables

$HTTP_HOST: nom d'hôte de la machine du client

$HTTP_REFERER: URL de la page qui a appelé le script PHP (cette variable permet de connaître le site web qui a renvoyé le client sur la page en cours: le moteur de recherche utilisé par exemple)

$HTTP_USER_AGENT: permet d'avoir des informations sur le type de navigateur utilisé par le client, ainsi que son système d'exploitation.

$REMOTE_ADDR: contient l'adresse IP du client appelant le script

$DOCUMENT_ROOT: chemin de la page en cours sur le serveur

$SERVER_ADDR: contient l'adresse IP du serveur

$DATE_GMT: date et heure actuelle au format GMT

$DATE_LOCAL: date et heure actuelle au format local

Pour afficher la valeur de la variable d'environnement:

echo getenv(''nom_de_la_variable'')

Les constantes PHP

Quelques constantes prédéfinies du PHP :

__FILE__ : nom du fichier en cours

__LINE__ : numéro de ligne en cours

PHP_VERSION : version de PHP

PHP_OS : nom du système d'exploitation qui est utilisé par la machine qui fait tourner le PHP

TRUE : la valeur vraie booléenne

FALSE : la valeur faux booléenne

Exemples :

```
$test = true;
echo __file__, __line__;
```

Les fichiers I

Les fichiers en PHP sont très souvent utilisées pour stocker les données. Les bases de données sont encore plus puissantes à ce niveau, mais il est parfois ennuyeux de les utiliser pour accueillir certaines formes de données. Exemple : un compteur de visite

Il existe une multitude de fonctions dédiées à l'utilisation des fichiers.

La plus simple est la fonction file_get_contents(). Elle prend comme paramètre le nom du fichier et elle affecte son contenu à une variable type chaîne de caractère.

Exemple:

Le fichier test.txt contient le texte suivant: Salut, c'est un test…

Pour afficher le contenu de ce fichier sur une page web:

```
$str=file_get_contents(''test.txt'');
echo $str; // affiche: Salut, c'est un test…
```

Les fichiers II

La manipulation de fichier se fait grâce à un identifiant de fichier.

Quelques fonctions:

fopen($file ,$mode) : ouverture du fichier identifié par son nom $file et dans un mode $mode particulier, retourne un identificateur de fichier $fp ou FALSE si échec

fclose($fp) : ferme le fichier identifié par le $fp

fgets($fp, $length) : lit une ligne de $length caractères au maximum. Si la longueur de la ligne est inférieur à $length, lit jusqu'à la fin de la ligne.

fputs($fp, $str) : écrit la chaîne $str dans le fichier identifié par $fp

fgetc($fp) : lit un caractère

feof($fp) : teste la fin du fichier retourne TRUE ou FALSE

file_exists($file) : indique si le fichier $file existe dans le dossier courant. Retourne TRUE ou FALSE. Le nom du fichier peut être une arborescence de répertoire.

Les fichiers III

Quelques fonctions - suite - :

filesize($file) : retourne la taille du fichier $file en octet

filetype($file) : retourne le type du fichier $file

unlink($file) : détruit le fichier $file

copy($source, $dest) : copie le fichier $source vers $dest

readfile($file) : affiche le fichier $file

rename($old, $new) : renomme le fichier $old en $new

fseek($fp,$pos) : place le pointeur du fichier identifié par $fp à la position $pos.

Modes d'ouverture de fichiers:

'r' lecture seule

'r+' lecture et écriture

'w' création et écriture seule

'w+` création et lecture/écriture

Les fichiers IV

Exemple typique d'affichage du contenu d'un fichier :

```
<?php
$file = ''fichier.txt'' ;
if( $fd = fopen($file, ''r'')) {          // ouverture du fichier en lecture
    while ( ! feof($fd) ) {               // teste la fin de fichier
          $str .= fgets($fd, 1024);
    /* lecture jusqu'à fin de ligne où des 1024 premiers caractères */
    }
    fclose ($fd);                         // fermeture du fichier
    echo $str;                            // affichage
} else {
    die(''Ouverture du fichier <b>$file</b> impossible.'');
}
?>
```

Les fichiers V

La fonction fopen permet d'ouvrir des fichiers dont le chemin est relatif ou absolu. Elle permet aussi d'ouvrir des ressources avec les protocols HTTP ou FTP. Elle renvoie FALSE si l'ouverture échoue.

Exemples :

```
$fp = fopen(''../docs/rapport.txt'', ''r'');
$fp = fopen(''http://www.php.net/'',''r'');
$fp = fopen(''ftp://user:password@www.ensa.ac.ma/'', ''w'');
```

Dates et heures I

Les fonctions de dates et heures sont très utilisées sur internent. En PHP il existe plusieurs fonctions qui permettent de manipuler les dates.

Quelques fonctions :

date(‘’$format’’) : retourne une chaîne de caractères contenant la date et/ou l’heure locale au format spécifié

getdate() : retourne un tableau associatif contenant la date et l’heure

checkdate($month, $day, $year) : vérifie la validité d’une date

mktime ($hour, $minute, $second, $month,$day, $year) : retourne le timestamp UNIX correspondant aux arguments fournis c’est-à-dire le nombre de secondes entre le début de l'époque UNIX (1er Janvier 1970) et le temps spécifié

time() : retourne le timestamp UNIX de l’heure locale

Dates et heures II

Les formats pour date

d Jour du mois sur deux chiffres [01..31]
j Jour du mois sans les zéros initiaux
l Jour de la semaine textuel en version longue et en anglais
D Jour de la semaine textuel en trois lettres et en anglais
w Jour de la semaine numérique [0..6] (0: dimanche)
z Jour de l'année [0..365]
m Mois de l'année sur deux chiffres [01..12]
n Mois sans les zéros initiaux
F Mois textuel en version longue et en anglais
M Mois textuel en trois lettres
Y Année sur 4 chiffres
y Année sur 2 chiffres
h Heure au format 12h [01..12]
g Heure au format 12h sans les zéros initiaux

Dates et heures III

Les formats pour date - suite –

H Heure au format 24h [00..23]
G Heure au format 24h sans les zéros initiaux
i Minutes [00..59]
s Secondes [00.59]
a am ou pm
A AM ou PM
L Booléen pour savoir si l'année est bissextile (1) ou pas (0)
S Suffixe ordinal anglais d'un nombre (ex: *nd* pour 2)
t Nombre de jour dans le mois donné [28..31]
Z Décalage horaire en secondes [-43200..43200]

Dates et heures IV

Les clés du tableau associatif retourné par getdate

seconds : secondes
minutes : minutes
hours : heures
mday : jour du mois
wday : jour de la semaine, numérique
mon : mois de l'année, numérique
year : année numérique
yday : jour de l'année numérique
weekday : jour de la semaine, textuel complet en anglais
month : mois textuel complet en anglais

Les expressions régulières I

Les expressions régulières sont un outil puissant pour la recherche de motifs dans une chaîne de caractères.

Fonctions

ereg($motif, $str) : teste l'existence du motif $motif dans la chaîne $str

ereg_replace($motif, $newstr, $str) : remplace les occurrences de $motif dans $str par la chaîne $newstr

split($motif, $str) : retourne un tableau des sous-chaînes de $str délimitées par les occurrences de $motif (semblable à explode())

Les fonctions eregi, eregi_replace et spliti sont insensibles à la casse (c'est-à-dire ne différencient pas les majuscules et minuscules).

Exemple

```
if (eregi(''Paris'', $adresse))
    echo ''Vous habitez Paris.'';
```

Les expressions régulières II

Les motifs peuvent être très complexes et contenir des caractères spéciaux.

Les caractères spéciaux

[abcdef] : intervalle de caractères, teste si l'un d'eux est présent

[a-f] : plage de caractères, teste la présence de l'un des caractères compris entre a et f

^ : le motif suivant doit apparaître en début de chaîne

\^ : recherche du caractère '^' que l'on déspécialise par l'antislash

. : remplace un caractère

* : remplace 0 ou plusieurs caractères à base du motif précédent

+ : remplace 1 ou plusieurs caractères à base du motif précédent

$: le motif précédant doit apparaître en fin de chaîne

Les expressions régulières III

Exemples de motifs

‘’[A-Z]’’ : recherche toutes les majuscules

‘’[a-zA-Z]’’ : recherche toutes les lettres de l’alphabet minuscules ou majuscules

‘’^Le ’’ : toute chaîne commençant par le mot ‘’Le ‘‘ suivi d’un espace

‘’\.com$’’ : toute chaîne se terminant par ’’.com’’ (déspécialise le point)

Exemples :

```
if ( ereg(‘’^.*@wanadoo\.fr’’, $email) ) {
    echo ‘’Vous êtes chez Wanadoo de France Télécom.’’;
}
```

Les expressions régulières IV

Il existe des séquences types :

[[:alnum:]] : [A-Za-z0-9] – caractères alphanumériques

[[:alpha:]] : [A-Za-z] – caractères alphabétiques

[[:digit:]] : [0-9] – caractères numériques

[[:blank:]] : espaces ou tabulation

[[:xdigit:]] : [0-9a-fA-F] – caractères hexadécimaux

[[:graph:]] : caractères affichables et imprimables

[[:lower:]] : [a-z] – caractères minuscules

[[:upper:]] : [A-Z] – caractères majuscules

[[:punct:]] : caractères de ponctuation

[[:space:]] : tout type d'espace

[[:cntrl:]] : caractères d'échappement

Cryptage de données

crypt($str,$salt) : Retourne une chaîne de caractères. Crypte la chaîne de caractères $str. La chaîne optionnelle $salt sert de base au cryptage. Cet argument optionnel est appelé « *grain de sel* ». L'algorythme de cryptage utilisé par PHP n'est a priori pas défini (il peut varier d'un système à un autre), mais c'est en général le DES standard ou bien encore MD5. Le système utilisé est injectif, c'est-à-dire qu'il n'existe pas d'algorithme symétrique pour décrypter la chaîne codée résultante.

md5($str) : Calcule et retourne la valeur MD5 de la chaine $str

md5_file($file): calcule et retourne la valeur MD5 du fichier $file

crc32($str) : calcule et retourne la somme de vérification de redondance cyclique (32 bit) de la chaîne $str. Cette valeur sert généralement à vérifier l'intégrité de données transmises.

Exemple:

```
$pass_crypt=md5($_POST[''pass'']); //code en md5 le mot de passe
                                   //passé en formulaire par POST.
```

Chargement de fichiers I

Les formulaires permettent de transmettre des informations sous forme de chaînes de caractères. Ils peuvent aussi permettre à un internaute de transmettre un fichier vers le serveur.

La balise HTML suivante : <input type=‘’file’’> permet le chargement de fichiers. La balise FORM doit nécessairement posséder l’attribut ENCTYPE de valeur ‘’multipart/form-data’’. La méthode utilisée sera POST. De plus, il est utile d’imposer au navigateur une taille de fichier limite par le paramètre MAX_FILE_SIZE dont la valeur numérique a pour unité l’octet.

Exemple

```
<form action=‘’index.php’’ method=‘’POST’’ ENCTYPE=‘’multipart/form-data’’>
<input type=‘’hidden’’ name=‘’MAX_FILE_SIZE’’ value=‘’1024000’’>
<input type=‘’file’’ name=‘’mon_fichier’’><br>
<input type=‘’submit’’ value=‘’envoyer’’>
</form>
```

Chargement de fichiers II

Pour récupérer le fichier, il faut utiliser la variable d'environnement $HTTP_POST_FILES qui est un tableau associatif dont les champs sont les noms des champs HTML file du formulaire. Par exemple : $HTTP_POST_FILES[‘mon_fichier‘] où mon_fichier est le nom donné au champs du formulaire HTML de type file.

La variable $HTTP_POST_FILES[‘mon_fichier‘] est elle aussi un tableau associatif possédant les champs suivants :

- name: nom du fichier chez le client
- tmp_name: nom temporaire du fichier sur le serveur
- size: taille du ficher en octets
- type: type mime du ficher (txt, jpeg…)

Exemple:

$nom_fichier = $HTTP_POST_FILES[‘’nom_fichier’’][‘’name’’];

Remarque:

Pour la version 4 et 5 du PHP la syntaxe $HTTP_POST_FILES[] peut se réduire à $_FILES[]

Chargement de fichiers III

Il est important de vérifier avec is_file() (ou file_exists())si un fichier du même nom existe déjà sur le serveur à l'emplacement où on veut copier le nouveau fichier. On pourra supprimer l'ancien fichier avec unlink() (qui ne fonctionne pas avec tous les serveurs fonctionnant sous Windows).

Même si le paramètre MAX_FILE_SIZE est inclus dans le formulaire, il est important de vérifier la taille du fichier réceptionné.

La vérification du type MIME du fichier est également importante dans le cas où on ne souhaite réceptionner que des types de fichiers bien particuliers (des images GIF, JPEG ou PNG par exemple).

Chargement de fichiers IV

La durée de vie du fichier temporaire sur le serveur est limitée au temps d'exécution du script. Pour pouvoir exploiter ultérieurement le fichier sur le serveur, il faut le sauvegarder dans un répertoire et lui donner un vrai nom.

Si aucun fichier n'a été envoyé par le client, la variable $mon_fichier vaudra la chaîne de caractères : ''none'' ou bien "" (chaîne vide).

Exemple:

```
<input type=''file'' name=''fichier''>
<input type=''hidden''  name=''max_file_size'' value=''1024000''>
<input type=''submit'' value=''Valider l'envoie du fichier''>
<?php
echo $HTTP_POST_FILES[''fichier''][''name''].''<br>'';
echo $HTTP_POST_FILES[''fichier''][''tmp_name''] .''<br>'';
echo $HTTP_POST_FILES[''fichier''][''size''] .''<br>'';
echo $HTTP_POST_FILES[''fichier''][''type''] .''<br>'';
?>
```

Chargement de fichiers V

On peut charger simultanément plusieurs fichiers en utilisant dans le formulaire HTML des champs file de noms différents.

Exemple :

```
<input type=‘’file’’ name=‘’mon_fichier’’>
<input type=‘’file’’ name=‘’mon_autre_fichier’’>
```

A la réception, chaque fichier est traité individuellement

Exemple :

```
$file = $HTTP_POST_FILES[‘mon_fichier’];
// puis faire le traitement vu précédemment
$file = $HTTP_POST_FILES[‘mon_autre_fichier’];
// puis refaire encore le même traitement
```

Mail I

La fonction mail() envoie un message électronique.

Syntaxe :

mail($destinataire, $sujet, $message, $entêtes);

Exemple :

$message = ‘’Rendez-vous à l’ENSA de Marrakech Mardi 5 Juin à 9h.’’;

mail(‘’contact@ensa.ac.ma’’, ‘’Rendez-vous’’, $message);

Les arguments obligatoires sont le destinataire, le sujet du message ainsi que le message proprement dit. Les entêtes et paramètres additionnels sont facultatifs.

Note: cette fonction ne marche que si un programme de messagerie électronique (appelé « mailer ») est préalablement installé sur le serveur.

Mail II

Exemple

```
<?php
$destinataire = ‘’etudiants <etudiants@ensa.ac.ma>, ‘’;
$destinataire.= ‘’enseignants<enseignant@ensa.ac.ma>’’;
$sujet = ‘’La Semaine Nationale de la Science’’;
$message = ‘’La SNS 2007 est prévue la première semaine de Mai’’;
$message .= ‘’Cette date n’est pas définitive’’;
$entêtes = ‘’From: ucam <contact@ucam.ac.ma>’’;
mail($destinataire, $sujet, $message, $entêtes);
?>
```

La fonction mail() renvoie true si l’envoie du message a été effectué avec succès sinon elle renvoie false.

Mail III

Quelques entêtes :

From: contact@ensa.ac.ma
X-Mailer: PHP\n // maileur
X-Priority: 1\n // Message urgent!
Return-Path: deamon@php-help.com // @ retour pour erreurs
Cc: archives@php-help.com // Champs CC
Reply-to: <hugo@php-help.com> // @ de retour

Format général des entêtes :
Nom-Entete: valeur

MCours.com