le cnam

PHP
Fonctions, librairie standard : fichiers, upload de fichiers, chaînes, dates

J-F Berger

P. Graffion

Novembre 2011

Fonctions (1)

- On peut définir ses propres fonctions
- On les déclare à l'aide de l'instruction function

```
function nom_fonction(paramètres) {
Corps de la fonction
}
```

- On ne précise ni le type des paramètres, ni la valeur de retour!
- Exemple:

```
function min($n1, $n2) {
   return ( $n1 < $n2) ? $n1 : $n2;
}
$min = min(3, 2); // => $min vaut 2
```

Fonctions (2)

- Par défaut, les arguments d'une fonction sont *transmis par valeur*.
- Exemple:

```
function inc($n1, $n2) {
  $n1++ ; $n2++;
}
$a = $b = 1;
inc($a,$b); // => $a et $b valent toujours 1!
```

Fonctions (3)

- Si un argument est *transmis par référence*, toute modification apportée à la variable paramètre modifie la variable de l'instruction d'appel.
- Une esperluette (&) est placée avant le nom du paramètre afin d'indiquer que l'argument doit être transmis par référence.
- Exemple:

```
function inc($n1, &$n2) {
  $n1++ ; $n2++;
}
$a = $b = 1;
inc($a,$b); // => $a vaut toujours 1, $b vaut 2!
```

Fonctions (4)

- On peut donner une valeur par défaut aux arguments lors de la déclaration de la fonction.
- Il suffit d'affecter une valeur au paramètre au moment de la déclaration.
- Tous les paramètres possédant des valeurs par défaut doivent figurer à droite des paramètres qui en sont dépourvus.
- Exemple:

```
function printRow($article, $style = ‘tdblue’) {
  …
}
printRow($article1); // ⇔ printRow($article1, ‘tdblue’);
```

Inclusion de fichiers php

La fonction **include()** inclut et interprète le fichier spécifié en argument

Permet d'importer :

- du code de librairies non-standard:

```
<?php
include 'myurlfunctions.inc.php';
$editParam = getHttpParameter('edit', '1');
?>
```

- des données:

```
<?php
include '../sql/mysqlparams.conf.php';
global $g_Host, $g_User, $g_Password;
mysql_connect($g_Host, $g_User, $g_Password);
?>
```

- des éléments de présentation:

```
<?php
include 'header.inc.php';
// …
include 'footer.inc.php';
?>
```

Portée des variables

- **La portée d'une variable diffère selon son emplacement** au sein du code PHP.
- **Les variables peuvent avoir une portée globale si elles sont définies en dehors d'une fonction.** Sinon, elles seront locales.

```
<?php   $a = 1; /* portée globale */
    function test() {
        echo $a; /* portée locale */}
    test();
    ?>   l'appel de test() n'affiche RIEN
```

- **Les fichiers inclus** dans un autre peuvent non seulement **accéder aux variables globales** définies dans la page hôte, mais aussi **les modifier.**
- **Les variables globales peuvent être utilisées dans une fonction** à condition de déclarer à nouveau les variables dans la fonction avec le mot-clé *global* ou en utilisant le tableau associatif prédéfini *$GLOBALS*.

Si on a :

```
function test()
{ global $a; echo $a; }  ou  { echo $GLOBALS["a"]; }
```

l'appel de test() affiche **1**

Fichiers (1)

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 $fp de fichier ou FALSE si échec
- fclose($fp) : ferme le fichier identifié par le $fp
- fgets($fp, $length) : lit une ligne de $length caractères au maximum
- fgetcsv ($fp) : Renvoie la ligne courante et cherche les champs CSV
- fputs($fp, $str) : écrit la chaîne $str dans le fichier identifié par $fp
- fgetc($fp) : lit un caractère
- feof($fp) : teste la fin du fichier
- file_exists($file) : indique si le fichier $file existe
- filesize($file) : retourne la taille du fichier $file
- filetype($file) : retourne le type du fichier $file
- unlink($file) : détruit le fichier $file
- copy($source, $dest) : copie le fichier $source vers $dest
- rename($old, $new) : renomme le fichier $old en $new

Fichiers (2)

La fonction **fopen** permet d'ouvrir des :

- fichiers dont le chemin est relatif ou absolu
- des ressources avec les protocoles HTTP ou FTP

Elle renvoie un identificateur $fp ou FALSE si l'ouverture échoue.

Exemples :

```
$fp = fopen("../docs/faq.txt", "r");
$fp = fopen("http://www.php.net/","r");
$fp = fopen("ftp://user:password@cia.gov/", "w");
```

Les modes d'ouverture :

'**r**' (lecture seule), '**r+**' (lecture et écriture), '**w**' (création et écriture seule), '**w+**' (création et lecture/écriture), '**a**'(création et écriture seule ; place le pointeur de fichier à la fin du fichier), '**a+**' (création et lecture/écriture ; place le pointeur de fichier à la fin du fichier)

Fichiers (3)

Exemple de lecture

```
<?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.");
}
?>
```

Parcours de répertoire (1)

Il est possible de parcourir les répertoires en utilisant :

- **chdir($str)** : Change le dossier courant en **$str**. Retourne TRUE si succès, sinon FALSE.
- **getcwd()** : Retourne le nom du dossier courant (en format chaîne de caractères).
- **opendir($str)** : Ouvre le dossier **$str**, et récupère un pointeur **$d** dessus si succès, FALSE sinon et génère alors une erreur PHP qui peut être échappée avec @.
- **closedir($d)** : Ferme le pointeur de dossier **$d**.
- **readdir($d)** : Lit une entrée du dossier identifié par **$d**. C'est-à-dire retourne un nom de fichier de la liste des fichiers du dossier pointé. Les fichiers ne sont pas triés. Ou bien retourne FALSE s'il n'y a plus de fichier.
- **rewinddir($d)** : Retourne à la première entrée du dossier identifié par **$d**.

Parcours de répertoire (2)

Exemple:

```
<?php
if ($dir = @opendir('.')) {                    // ouverture du dossier
    while($file = readdir($dir)) {     // lecture d'une entrée
        echo "$file<br />";          // affichage du nom de fichier
    }
    closedir($dir);                            // fermeture du dossier
}
?>
```

$dir est un pointeur vers la ressource dossier

$file est une chaîne de caractères qui prend pour valeur chacun des noms de fichiers retournés par **readdir()**

Upload de fichier (1)

Le protocole HTTP permet à l'utilisateur de télécharger des fichiers sur le serveur via un formulaire HTML.

Ce formulaire doit contenir :

- Un attribut supplémentaire dans la balise <form> : enctype="multipart/form-data"
- autant de balises <input type="file"> que de fichiers à télécharger.
- éventuellement, une balise qui précise la taille maximale du fichier accepté, en octets (à placer avant la balise file)
  <input type="hidden" name="MAX_FILE_SIZE" value="xxx">,

```
<form name="editMember2Action.php" action="upload.php"
      method="POST" enctype="multipart/form-data">
Votre Photo : <input type="file" name="photo">
<br><input type="submit" value="envoyer">
</form>
```

Upload de fichier (2)

Une fois le formulaire posté :

- les fichiers indiqués par l'utilisateur sont téléchargés sur le serveur et enregistrés dans un répertoire temporaire (indiqué dans php.ini).
- la page php indiquée dans l'action du formulaire s'exécute,
- elle consulte la variable globale $_FILES : un tableau associatif contenant toutes les informations des fichiers téléchargés, chaque fichier s'obtenant en utilisant pour clé le nom de la balise file

Exemple:

- $_FILES[**'photo'**]['name'] :
  le nom du fichier stocké sur la machine du client qui envoie le fichier.
- $_FILES[**'photo'**]['tmp_name'] :
  le nom du fichier stocké sur le serveur (le fichier a donc été reçu).
- $_FILES[**'photo'**]['type'] :
  le type mime du fichier envoyé. Attention aux failles de sécurité.
- $_FILES[**'photo'**]['size'] :
  la taille du fichier en octets.
- $_FILES[**'photo'**]['error'] :
  éventuellement un code d'erreur pendant le transfert.

Upload de fichier (3)

L'étape suivante consiste à copier le/les fichiers téléchargés dans un répertoire définitif (après en avoir vérifié la taille par filesize()):

```
if (is_uploaded_file($_FILES['photo']['tmp_name'])) {
    copy($_FILES['photo']['tmp_name'],
          "/chemin/destination/".$_FILES['photo']['name']);
} else {
    echo "attaque upload !!! fic=".$_FILES['photo']['name'];
}
```

ou

```
move_uploaded_file($_FILES['photo']['tmp_name'],
         "/chemin/destination/".$_FILES['photo']['name']);
```

Traitement des chaînes de caractères (1)

BUTS :

- Formater des chaînes.
- Assembler, découper des chaînes.
- Comparer des chaînes.
- Reconnaître et remplacer des sous-chaînes.

Outils :

- Plus de 50 fonctions standard en PHP
- Les Expressions régulières (POSIX et aussi PERL)

Traitement des chaînes de caractères (2)

- Rappels, les chaînes sont représentées entre
  - Guillemets simples : pas d' interprétation sauf \\ et\' pour \ et ‘
  - Guillemets doubles : interprétation des variables et des caractères \n, \t, \r, \$, \\ et \"
- Affichage par echo ou print
- ATTENTION au mélange PHP-HTML :
  - Exemple \n = saut de ligne dans le code généré mais pas en html…(
)

Chaînes (1) - Découpage

- **explode($sep, $str [, $n])** : scinde la chaîne **$str** en au plus **$n** morceaux en utilisant le séparateur **$sep**. Retourne un tableau.
- **implode([$sep,] $tab)** : fonction réciproque de **explode()**. Chaîne composée des éléments de $tab séparés par *$sep [ou collés si $sep* absent]

Exemple :

```
<?php
$string="un-deux-trois;";
print_r($tableau  = explode('-',$string));
echo "<br />".implode('-',$tableau) ;
?>
```

Génère :

```
Array ( [0] => un [1] => deux [2] => trois; )
un-deux-trois;
```

Chaînes (2) - Substitution

autres exemples :

- substr(*$string, $start, $length*) renvoie la sous-chaîne de longueur *$length*, depuis le ,*$start* ème caractère. Si $start est négatif alors $start est évalué à partir de la fin.
- addslashes(*$string)* :ajoute(enlève) **** devant les caractères spéciaux (les guillemets simples ('), guillemets doubles ("), anti-slash (\ ) et NUL (le caractère NULL)) dans *$string* (pb pour les requêtes sql…)
- stripslashes(*$string*) : l'inverse

```
<?php
$str= addslashes('et "voila" \$donc'."  l'autre \$ ?<br />");
echo "$str  <br />";
echo stripslashes($str);
?>
et \"voila\" \\$donc l\'autre $ ?
et "voila" \$donc l'autre $ ?
```

- D'autres fonctions permettent d'enlever les caractères blanc,\r,\n,\t,…

Chaînes(3) - str_replace

Fonction plus complexe :

- str_replace*($search,$replace,$string*) : remplace les occurrences de *$search* dans *$string* par *$replace*…

En fait tous les paramètres de str_replace peuvent être des tableaux :

- mixed **str_replace** (mixed search , mixed replace , mixed string)
  - Si string est un tableau, alors le remplacement se fera sur chaque élément de celui-ci, et la valeur retournée sera aussi un tableau.
  - Si search et replace sont des tableaux, alors str_replace prendra une valeur de chaque tableau, et l'utilisera pour faire le remplacement dans string. Si replace a moins de valeurs que search, alors une chaîne vide sera utilisée pour effectuer les remplacements. Si search est un tableau et que replace est une chaîne, alors la chaîne de remplacement sera utilisée pour chaque élément de search. (ouf!)

Chaînes(4) - str_replace

Exemples str_replace **:**

```
// Génère : <body text='black'>
$string = str_replace("%body%", "black", "body text='%body%'");
echo $string ; echo "<br />";

$voyelles = array("a", "e", "i", "o", "u", "A", "E", "I", "O", "U");
echo $consonnes = str_replace($voyelles, "", "Bonjour le monde" );
echo "<br />";
$phrase  = "Ah manger des fruits, des légumes et des fibres ...";
$regime = array("fruits", "légumes", "fibres");
$bonne_chere = array("pizzas", "bières" , "gâteaux");
echo $newphrase = str_replace($regime, $bonne_chere, $phrase);
echo "<br />";
?>
```

Chaînes (5) - Trimming

- string **trim** (string str , *[string charlist]*) supprime des caractères invisibles en début et fin de chaîne. Si charlist a été omis, suppression des caractères :
  - " " (ASCII 32 (0x20)), "\t" (ASCII 9 (0x09)), "\n" (ASCII 10 (0x0A)), une nouvelle ligne , "\r" (ASCII 13 (0x0D)), "\0" (ASCII 0 (0x00)), le caractère NUL , "\x0B" (ASCII 11 (0x0B)), tabulation verticale.
  - charlist spécifie les caractères à supprimer.
- **chop, rtrim** : supprime les blancs en fin
- **ltrim** : supprime les blancs en début

Chaînes (6) - Divers

- **nl2br** : change les caractères de passage à la ligne en tag break de html (
 pour la compatibilité avec xHTML).

- **strtoupper** : passe la chaîne en capitales
- **strtolower** : passe la chaîne en minuscules

- **ucfirst** : passe la 1ère lettre de la chaîne en capitale
- **ucwords** : met en majuscule la 1ère lettre de tous les mots, si ce caractère est alphabétique.

Chaînes (7) - Comparaison

- **strlen** : nombre de caractères dans la chaîne
- **==**
- **strcmp** : compare 2 chaînes
  - retourne 0 si égale, > 0 si la 1ère est supérieure, < 0 si la 2ème est supérieure
  - Sensible à la casse
- **strcasecmp** : comme strcmp mais insensible à la casse.
- **strnatcmp** : comparaison selon “l’ordre naturel” (lexicographique cf. www.naturalordersort.com)

Chaînes (8) - Recherche

- **strstr** ou **strchr**

string **strstr** (string haystack , string needle)

retourne la sous-chaîne de haystack , depuis la première occurrence de needle (comprise) jusqu'à la fin de haystack. Si needle est introuvable, retourne FALSE . Si needle n'est pas une chaîne, elle est convertie en entier, et utilisée comme code ascii du caractère correspondant.

Sensible à la casse.

Chaînes (9) - Recherche

stristr : cf. strstr mais NON sensible à la casse

strpos : retourne la position numérique de la première occurrence du 2ème param.

strrchr : comme strstr mais à partir du début de la dernière occurrence

strrpos : retourne la position numérique de la dernière occurrence du 2ème param.

Chaînes (9) - Recherche

- **strtr** :

string **strtr** (string str , string from , string to) retourne la chaîne str , après avoir remplacé chaque caractère du paramètre from par le caractère de même position dans le paramètre to . Si from et to sont de tailles différentes, les caractères en trop dans l'un ou l'autre seront ignorés.

string **strtr** (string str , array from)

Avec 2 arguments : from doit alors être un tableau associatif de paires, qui seront remplacées dans la chaîne str. strtr commence toujours par rechercher la chaîne la plus longue, et ne travaille pas sur des segments qu'elle a déjà modifié.

Chaînes (10) …

- Voir en annexe
- Ou bien en : http://www.manuelphp.com/php/strings.intro.php
- Ou en http://fr.php.net/addslashes
- Ou en …

Dates (1)

timestamp UNIX d'une date = nombre de secondes entre le début de l'époque UNIX (1er Janvier 1970) et cette date

- **time**() : retourne le timestamp UNIX actuel
- **date**($format, $timestamp = time()) : retourne une chaîne de caractères contenant la date et/ou l'heure locale au format spécifié
- **getdate**($timestamp = time()) : retourne un tableau associatif contenant la date et l'heure
- **mktime** ($hour, $minute, $second, $month,$day, $year) : retourne le timestamp UNIX de la date précisée

Dates (2)

Formats pour la fonction **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

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 bisextile (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]

U Secondes depuis une époque **Z** Décalage horaire en secondes [-43200..43200]

Dates (3)

Exemple 1 :

- echo date("Y-m-d H:i:s");
- /* affiche la date au format MySQL : '2006-06-18 22:30:29' */

Exemple 2 :

- $aujourdhui = getdate();
- $mois = $aujourdhui['mon'];
- $jour = $aujourdhui['mday'];
- $annee = $aujourdhui['year'];
- echo "$jour/$mois/$annee"; // affiche ‘18/6/2006'