PHP – Les bases – Partie 3

Les include et les require

L'instruction de langage ***include()*** inclut et exécute le fichier spécifié en argument.

Les fichiers sont inclus suivant le chemin du fichier fourni ; si aucun n'est fourni, l'include_path sera vérifié.

Les "include_path" sont un ou des chemins réglés sur le serveur (dans php.ini) dans lesquels certains scripts PHP vont cherchés les fichiers appelés.

Si le fichier n'est pas trouvé dans l' include_path, *include* vérifiera dans le dossier du script appelant et dans le dossier de travail courant avant d'échouer.

L'instruction *include* enverra une erreur de type warning si elle ne peut trouver le fichier; ce comportement est différent du ***require()***, qui enverra une erreur de niveau fatal, ce qui arrêtera donc d'interprété le code dès l'erreur.

```
<?php

// importe le fichier header.php , affiche une erreur si il n'est pas trouvé, mais continue à exécuter le script
include('header.php') ;

echo 'coucou' ; // sera affiché

// importe le fichier config.php , affiche une erreur si il n'est pas trouvé, le script s'arrête ici
require('config.php') ;

echo 'les amis' ; // ne sera pas affiché

?>
```

Lorsqu'un fichier est inclus, le code le composant hérite de la portée des variables de la ligne où l'inclusion apparaît.

Toutes les variables disponibles à cette ligne dans le fichier appelant seront disponibles dans le fichier appelé, à partir de ce point.

Cependant, toutes les fonctions et classes définies dans le fichier inclus ont une portée globale.

```php
<?php

// contenu de config.php

$var1 = 5 ;
$var2 = 'coucou' ;

// contenu de monfichier.php

require('config.php') ;

echo $var1.' '.$var2 ; // affiche 5 coucou

?>
```

L'instruction ***require_once()*** est identique à require mis à part que PHP vérifie si le fichier a déjà été inclus, et si c'est le cas, ne l'inclut pas une deuxième fois.

L'utilisation la plus courante de cette fonction est lorsque vous avez plusieurs fonctions définies dans le fichier qui sera inclus.

Pour éviter d'avoir des erreurs sur la redéfinition de fonctions, on utilise include_once ou require_once pour n'inclure le fichier qu'une seule et unique fois.

On les utilisera souvent dans les CMS et les frameworks, car cela permet un compatibilité du code optimal car il est créé par une multitude de développeurs.

Il en est de même pour ***include_once()***.

```php
<?php

	include('include.php');

	  echo $var.' '; // 5
	  $var++;
	  echo $var.' '; // 6

	include 'include.php'; // autre écriture

	  echo $var.' '; // 5
	  $var++;
	  echo $var.' '; // 6

	include_once('include.php');
	  echo $var.' '; // reste 6 car non inclus

?>
```

Les fonctions utilisateurs

Une fonction est ce que l'on peut appeler un sous programme, une procédure.

On distingue deux types de fonctions : les "**fonctions intégrées**" ou "**built-in**" qui sont incluses par défaut avec les distributions de **PHP** comme **print**, **echo** (les fonctions génériques telles que celles-ci peuvent également être nommées « **structure du langage** ») et les fonctions définies par le programmeur, dites aussi "**fonctions utilisateur**".

Les fonctions ont plusieurs buts:

- Éclaircir le code en regroupant certaines fonctionnalités d'un programme qui se répètent dans une même fonction.

- Pouvoir créer des fonctions génériques qui pourront être utilisées dans d'autres programmes, ce qui évite de répéter pour chaque projet le même code.

- Possibilité d'évolution du code plus facile dans la mesure où lorsque vous modifiez le contenu d'une fonction, les répercussions sont effectuées sur l'ensemble du programme sans que vous ayez à le modifier dans la plupart des cas (sauf si vous rajoutez des paramètres etc).

Déclaration d'une fonction utilisateurs

PHP recèle de nombreuses fonctions intégrées permettant d'effectuer des actions courantes. Toutefois, il est possible de définir des fonctions, dites *fonctions utilisateurs* afin de simplifier l'exécution de séries d'instructions répétitives.

Contrairement à de nombreux autres langages, PHP nécessite que l'on définisse une fonction avant que celle-ci puisse être utilisée (sauf en cas de bufférisation ou mise en cache serveur, que nous verrons plus tard).

Pour l'appeler dans le corps du programme il faut que l'interpréteur la connaisse.

C'est-à-dire qu'il connaisse son nom, ses arguments et les instructions qu'elle contient.

La définition d'une fonction s'appelle "*déclaration"* et peut se faire n'importe où dans le code.

Une fonction utilisateur est définie par deux principaux éléments. Sa signature et son corps. La signature est elle même composée de deux parties : le nom et la liste de paramètres.

Le corps, quant à lui, établit la suite d'instructions qui devront être exécutées.

La déclaration d'une fonction se fait grâce au mot-clé *function*, selon la syntaxe suivante :

```
<?php
function Nom_De_La_Fonction(argument1, argument2, ...) {
   liste d'instructions
}
?>
```

Remarques:

- le nom de la fonction suit les mêmes règles que les noms de variables
- le nom doit commencer par une lettre
- un nom de fonction peut comporter des lettres, des chiffres et les caractères _ et & (les espaces ne sont pas autorisés!)
- le nom de la fonction, comme celui des variables est sensible à la casse (différenciation entre les minuscules et majuscules)
- les arguments sont facultatifs, mais s'il n'y a pas d'arguments, les parenthèses doivent rester présentes
- les arguments peuvent avoir une valeur par défaut (contrairement au javascript où il faut "tricher" pour obtenir le même résultat
- il ne faut pas oublier de refermer les accolades
- le nombre d'accolades ouvertes (fonction, boucles et autres structures) doit être égal au nombre d'accolades fermées!
- la même chose s'applique pour les parenthèses, les crochets ou les guillemets!

Une fois cette étape franchie, votre fonction ne s'exécutera pas tant que l'on ne fait pas appel à elle quelque part dans la page!

```
<?php
ma_fonction();
?>
```

Il existe deux types de fonctions utilisateurs : les **fonctions** et les **procédures**.

Quelle différence y-a-t-il entre les deux ? La différence qui les sépare est la valeur de retour.

Une procédure est un groupe d'instructions qui ne renvoie pas de valeur après leur exécution.

Une fonction, quant à elle, est en réalité une procédure qui retourne une valeur de type primitif (souvant true ou false ou complexe) ou un objet. On utilise le **return** pour cela.

Enfin, une fonction utilisateur peut (ou non) prendre des paramètres d'entrée au même titre qu'une fonction mathématique. Ces paramètres peuvent être de type primitif (int, float, string, boolean) ou structuré (array, object).

```php
<?php

function coucou(){
            return 'yeah<br/>';
        }
        echo coucou();

        function coucou2($att){
            return 'yeah<br/>';
        }
        echo coucou2(); // erreur car argument manquant

        function coucou3($att,$att2){
            return "$att - $att2<br/>";
        }
        echo coucou3('un','deux'); // pas d'erreur

        function coucou4($att=1,$att2=2){
            return "$att - $att2<br/>";
        }
        echo coucou4(); // pas d'erreur et affiche les valeurs par
défaut
        echo coucou4(3,4); // pas d'erreur et changement des
valeurs par défaut

?>
```

Exemple de script qui crée une chaîne numérique de X caractères :

```php
<?php

function chaine($nb = 5){

            $sortie = "";

            for($i=0;$i<$nb;$i++){
                $sortie .= mt_rand(0, 9);
            }

            return $sortie;
        }

        echo chaine(); // affiche 5 caractères par défaut


        echo "<br/>";
        echo chaine(30); // affiche 30 caractères

?>
```

Pour passer plusieurs valeurs comme résultat d'une fonction, on peut les stocker dans un array

```php
<?php
// variables a traiter avec notre fonction
$mavar = 5;
$mavar2 = 3;


// function qui va renvoyer un tableau avec l'addition, la soustraction, la multiplication et la division des 2 paramètres
function calcul($a,$b){

    $tableau["+"]= $a+$b;
    $tableau["-"]= $a-$b;
    $tableau["*"]= $a*$b;
    $tableau["/"]= $a/$b;

    return $tableau; // le tableau associatif contenant les valeurs est envoyé grâce au return, le return arrête la fonction après sont envoi
}

var_dump(calcul($mavar,$mavar2));

// affiche :
array (size=4)
  '+' => int 8
  '-' => int 2
  '*' => int 15
  '/' => float 1.6666666666667

?>
```

Les fonctions imbriquées

Les fonctions peuvent être imbriquées, c'est à dire être encapsulées dans une ou des autres fonctions.

```php
<?php
// variables a traiter avec notre fonction
$mavar = " Il était une fois une famille pauvre qui comptait trois fils : Pierre, Jacques et Jean. Un jour, Pierre, l'aîné, dit : « Je vais aller chercher du travail ; je reviendrai quand je serai riche. » Il partit sur la grand-route et marcha, marcha. Un soir, n'ayant plus qu'un croûton de pain à se mettre sous la dent, il rencontra une vieille qui lui demanda ce qu'il désirait le plus.";


// fonction qui coupe notre chaine de caractère si plus longue que 100 (par défaut)
function resume($texte,$caracteres=100){

    if(strlen($texte)>100){ // si notre texte a plus de 100 caractères
        // on coupe la chaine après X caractères
        $texte = substr($texte, 0 , $caracteres);
        // on rajoute les '...'
        $texte .= " ... ";
    }
    // on renvoie le texte en protégeant au préalable celui-ci pour l'insertion dans une db
    return protegeDB($texte);

}

// fonction qui protège le texte pour l'insérer dans une DB
function protegeDB($texte){

    // on retire les espaces devant et derrière
    $texte = trim($texte);
    // on retire les tags
    $texte = strip_tags($texte);
    // on encode les caractères dangereurs en entités html
    $texte = htmlspecialchars($texte, ENT_QUOTES);
    // on renvoie le résultat
    return $texte;
}

echo resume($mavar,120);

/* affiche :
Il était une fois une famille pauvre qui comptait trois fils : Pierre, Jacques et Jean. Un jour, Pierre, l&#039;aîné, dit ...
*/
?>
```