Les variables dynamiques

Introduction

Les variables dynamiques sont un aspect de PHP peu connu par les débutants, vous n'en entendrez pas parler dans la plupart des bouquins traitant de PHP ou alors très brièvement, cependant elles peuvent s'avérer vraiment utiles dans certaines situations, et c'est cette utilité que nous tenterons de mettre à jour.

Notre premier exemple

Pour commencer prenons un exemple, tout ce qu'il y a de plus simple, c'est juste de mieux vous préparer à la suite :

```
<?php

$variable =  'nom';
$$variable =  'phpdeb';

?>
```

Vous vous demandez certainement à quoi peut nous servir ce code, supposons que vous vouliez afficher une variable choisie par le visiteur et passée par l'url (soit $_GET[]), vous devriez utiliser des if pour cerner toutes les possibilités, mais en utilisant cette astuce vous réduisez votre code à seulement quelques lignes et vous obtenu le résultat voulu tout en gardant un code propre et net. Notre visiteur clique sur un lien par exemple et il est mené à cette url : 'http://www.phpdebutant.org/test.php?variable=nom', il demande donc à voir le contenu de la variable nom, voyons comment procède le code de la page test.php :

```
<?php

// Pour cet exemple nous supposerons que PHP_VERSION => 4.1.0
$variable = $_GET['variable'];
$nom = 'phpdeb';

// affichage de la variable demandée par le visiteur
if($variable == 'nom'){
    echo $$variable;
}

// ce qui affiche 'phpdeb'
?>
```

Il existe deux façon de déclarer ou d'afficher les variables dynamiques, ainsi le code suivant aurait tout à fait le même résultat à l'affichage que celui cité plus haut :

```
<?php

// Pour cet exemple nous supposerons que PHP_VERSION => 4.1.0
$variable = $_GET['variable'];
$nom = 'phpdeb';

// affichage de la variable demandée par le visiteur
if($variable == 'nom'){
    echo ${$variable};
}

// ce qui affiche aussi 'phpdeb'.

?>
```

Il faut aussi savoir qu'il n'y a pas de limite à l'utilisation de ce type de variables, nous pourrions donc faire d'une variable dynamique une autre variable dynamique et ceci à l'infini, voyons un exemple :

```
<?php

$variable = $_GET['variable'];
$nom = 'phpdeb';
```

```php
$phpdeb = 'autre_var';
$autre_var = 'Et voilà le travail';

// affichage de la variable $autre_var
if($variable == 'nom'){
    echo $$$variable;
}

// ce qui va afficher 'Et voilà le travail'

?>
```

ATTENTION !!!

Ces exemples affichent des variables en fonction de ce que contient l'url. N'oubliez pas qu'il ne faut **jamais** se fier à des données provenant de GET (par l'url) ou de POST (par des formulaires). Il vous faut absolument contrôler le contenu de ces variables pour s'assurer qu'il n'y a pas de risque d'afficher des variables contenant des données sensibles. C'est ce qu'on a fait en ajoutant des if avant d'afficher. Cependant les variables dynamiques ne sont pas dangereuses en elles−même, c'est le fait d'utiliser une variable provenant de GET, le problème est le même avec les pseudos−frames.

Tableaux dynamiques

Il est aussi possible d'utiliser des noms dynamiques pour les variables tableaux, le fonctionnement ne change presque pas par rapport à ce qu'on a vu en haut pour les variables (sachant qu'un array() est une forme de variable étendue), voici un petit exemple d'utilisation :

```php
<?php

// Déclaration des différents array()
$pizzas = array('royale','vegetarienne');
$fruits = array('fraise','orange');

$variable = 'pizzas';

$i = 0;
while ($i < count($$variable)) {
    echo $$variable[$i];
    echo '<br/>';
    $i++;
}

?>
```

Ce code ne produit pas l'effet attendu, voici un screenshot du résultat obtenu, ça n'a vraiment rien à voir avec ce qu'on voulait, et c'est là que diffère l'utilisation des array :

Modification du code :

Le déboggage de ce code n'est pas très difficile, car la première chose qu'on essaie c'est de changer $variable en ${$variable} et on voit que ca marche bien, il est donc indispensable d'utiliser les accolades {} pour l'utilisation de tableaux dynamiques, le code devient donc :

```php
<?php

// Déclaration des différents array()
$pizzas = array('royale','vegetarienne');
$fruits = array('fraise','orange');

$variable = 'pizzas';

for ($i=0; $i < count(${$variable}); $i++) {
```

```
echo ${$variable}[$i].'<br/>';
}

?>
```

Dans ce cas, le script ne produira pas d'erreur et affichera bien le contenu du tableau :

Les fonctions dynamiques

Il est aussi possible aussi d'utiliser des fonctions dynamiques, c'est à dire que les fonctions sont déjà définies par exemple et l'on ne sait pas encore laquelle utiliser, pour éclaircir tout ça, rien de mieux qu'un exemple ;) :

```
<?php

// Recuperation de l'action demandée par l'url
$action = $_GET['action'];

function ajoute($x, $y) {
    return $x+$y;
}

function soustrait($x, $y) {
    return $x-$y;
}

// execution de la fonction
if($action == 'soustrait' or $action == 'ajoute'){
    echo $action('4', '2');
}

?>
```

Cette page test-maths.php si elle appelée de cette facon : 'http://localhost/test-maths.php?action=ajouter' retournera 6; bien sûr vous n'êtes toujours pas convaincus de l'utilité de ces variables, il faudra que vous y soyez confrontés lors d'un de vos projets, c'est là que vous verrez à quelle point elles sont parfois indispensables.

Exemple d'utilisation des fonctions dynamiques :

Pour mettre en évidence l'utilité des fonctions dynamiques nous allons considérer un cas spécial. Nous disposons d'un système basique de gestion de shopping cart, utilisé dans un site de e-commerce, le client a le choix entre cliquer sur 'plus' ou 'moins'. le 1er lien mène vers la page 'ajouter-qte-produit.php' dont voici le code :

```
<?php

$fonction = $_GET['fonction'];

// on suppose qu'une connexion à mysql est déjà ouverte ...

function plus() {
    global $membre;
    $sql = 'UPDATE shopping SET quantite=quantite+1 WHERE
membre='.$membre;
    mysql_query($sql);
}

function moins() {
    global $membre;
    $sql = 'UPDATE shopping SET quantite=quantite-1 WHERE
```

```
membre='.$membre;
}

if ($fonction == 'plus') {
    plus();
}elseif ($fonction == 'moins') {
    moins();
}


?>
```

Imaginez que vous ayez une dizaine d'actions possibles, il vous faudrait toutes les passer par les if, et cela pourrait s'avérer long et surtout facilement contournable, voici ce que serait le code si on utilisait les fonctions dynamiques:

```
<?php

$fonction = $_GET['fonction'];

function plus() {
    global $membre;
    $sql = 'UPDATE shopping SET quantite=quantite+1 WHERE
membre='.$membre;
    mysql_query($sql);
}

function moins() {
    global $membre;
    $sql = 'UPDATE shopping SET quantite=quantite-1 WHERE
membre='.$membre;
    mysql_query($sql);
}

// on doit quand même controler si c'est une fonction que l'on autorise
// à éxécuter, sinon gare à la sécurité
if($fonction == 'plus' || $fonction == 'moins'){
    $fonction();
}

?>
```

Les interdits

Comme toujours en PHP, il y a quelques limitations à l'utilisation de ces variables, la plus importante est qu'on ne peut pas 'coller' deux variables, le code suivant donnera un **parse error** :

```
<?php

$variable='pizza';
$action='acheter';

$acheter_pizza = 'miam';

// ceci est invalide
echo $action_$variable;

// il faut faire
echo ${$action.'_'.$variable};
```

```
?>
```
Ce cours touche maintenant à sa fin et j'espère qu'il vous aura aidé à mieux comprendre le fonctionnement souvent ignoré de ces variables.

HbiLLo