Apostrophes ou guillemets : lesquels choisir ?

par Pierre-Baptiste Naigeon

Date de publication : 12 Avril 2006

Dernière mise à jour : 02 Mai 2006

Utilisation d'une chaîne : avec des guillemets, avec des apostrophes ??? Lesquels choisir ? Nous allons voir ici les différences d'utilisation entre les deux, à vous après de faire votre choix en toute connaissance de cause.

Introduction

En php, lorsque l'on utilise une chaîne de caractère, on ne connait souvent pas la différence entre l'utilisation des apostrophes (') et les guillemets ("). Pourtant, il y a entre les deux une différence fondamentale...

I - Différence entre les deux :

Prenons ce bout de code :

Utilisation différente des deux
```
$i=42;

echo "Valeur de i : $i";
echo 'Valeur de i : $i';
```

Le premier echo va afficher : Valeur de i : 42

Le second va afficher : Valeur de i : $i

On comprend avec cet exemple que les chaînes entre guillemets savent interpréter les variables PHP présentes à l'intérieur, alors qu'une chaîne entre apostrophes va se contenter d'afficher 'bêtement' son contenu.

Pour aficher la valeur de i en utilisant les apostrophes, il va nous falloir utiliser l'opérateur de concaténation, le point (.).

Syntaxe correcte des apostrophes
```
echo 'Valeur de i : '.$i;
```

Maintenant, cela affichera : Valeur de i : 42

II - Lisibilité du Code :

Essayons maintenant d'afficher des balises HTML, avec différents attributs et une variable en utilisant les deux méthodes :

Affichage de balises HTML
```php
$i='titre du lien';

echo "<p><a href=\"#\" class=\"test\" onclick=\"rien();\">$i</a></p>";
echo '<p><a href="#" class="test" onclick="rien();">'.$i.'</a></p>';
```

Les fonctions et styles utilisés ici sont purement fictifs, mais ne remarquez-vous pas une énorme différence ? **La lisibilité du code.**

Dans le premier exemple, avant chaque guillemet présent dans la chaîne, il a fallu ajouter un backslash (\), afin d'empêcher PHP de considérer ce guillemet comme étant la fin de la chaîne.

Dans le deuxième cas, on a certes du rajouter l'opérateur de concaténation, mais le code est tout de même nettement plus clair.

III - Temps d'exécution :

Cette fonction sera utilisée dans tout les exemples suivants, pour mesurer le temps d'exécution du script :

Fonction de calcul du temps
```
//Fonction de calcul du temps :
function microtime_float() {
        list($usec, $sec) = explode(" ", microtime());
        return ((float)$usec + (float)$sec);
}
```

Effectuons un petit test sur un très grand nombre d'occurences avec les deux méthodes :

```
$debut_guillemet = microtime_float();
for ($i=0; $i<3000000; $i++) {
        $toto = "<a href=\"#\" class=\"test\" onclick=\"rien();\">$i</a>";
}
$fin_guillemet = microtime_float() - $debut_guillemet;

$debut_apostrophe = microtime_float();
for ($i=0; $i<3000000; $i++) {
        $toto = '<a href="#" class="test" onclick="rien();">'.$i.'</a>';
}
$fin_apostrophe = microtime_float() - $debut_apostrophe;

echo "<br>temps avec les guillemets : $fin_guillemet";
echo '<br>temps avec les apostrophes : '.$fin_apostrophe;
```

- temps avec les guillemets : 23.594511985779
- temps avec les apostrophes : 5.071653842926

Une différence significative **du simple à plus du quadruple !!!**

Certes, vous afficherez rarement 3 000 000 de lignes avec cette méthode sur une même page, mais dans le cas par exemple d'une extraction / affichage depuis une base de données, cela peut changer beaucoup de choses dans la durée d'affichage de la page ou dans le temps d'exécution du script.

Reprenons maintenant ce test avec le premier exemple (et oui, vous pouvez vous contenter d'afficher du texte sans balises HTML) :

```
$debut_guillemet = microtime_float();
for ($i=0; $i<3000000; $i++) {
        $toto = "Valeur de i : $i";
}
$fin_guillemet = microtime_float() - $debut_guillemet;

$debut_apostrophe = microtime_float();
for ($i=0; $i<3000000; $i++) {
        $toto = 'Valeur de i : '.$i;
}
$fin_apostrophe = microtime_float() - $debut_apostrophe;

echo "<br>temps avec les guillemets : $fin_guillemet";
echo '<br>temps avec les apostrophes : '.$fin_apostrophe;
```

- temps avec les guillemets : 9.5818469524384
- temps avec les apostrophes : 4.2532429695129

IV - Conclusion :

J'aurai tendance à privilégier l'utilisation des apostrophes, qui rendent le code plus lisible, et qui, cerise sur le gâteau, vont également accélérer l'affichage des pages :)

Mais il faut apprendre à faire preuve de discernement... Ainsi, dans le cas d'une requête SQL, l'écriture avec des guillemets reste plus claire :

Contre-exemple avec des requêtes SQL

```
$requete_guillemet  = "DELETE FROM $table_x WHERE id_y = '$id_z'";
$requete_apostrophe  = 'DELETE FROM '.$table_x.' WHERE id_y = \'' . $id_z . '\'';
```

Et qui dit code clair dit code plus facile à maintenir...

V - Annexe - Autres tests de temps d'exécution

Afin d'avoir en main toutes les clés pour pouvoir choisir en toute connaissance de cause, voici deux autres tests.

Dans les deux tests ci-dessous, la même fonction pour calculer le temps qu'à la section III a été utilisée.

Seul le nombre de passage dans la boucle a été augmenté, afin de rendre les tests plus significatifs.

Test d'affichage simple :

affichage classique
```
for ($i=0; $i<20000000; $i++) {
        $toto = "Valeur";
}

for ($i=0; $i<20000000; $i++) {
        $toto = 'Valeur';
}
```

- temps avec les guillemets : 10.029242992401
- temps avec les quotes : 10.030436038971

Test avec concaténation :

Concaténation simple
```
for ($i=0; $<10000000; $i++) {
        $toto = "Valeur de i : ".$i;
}

for ($i=0; $i<10000000; $i++) {
        $toto = 'Valeur de i : '.$i;
}
```

- temps avec les guillemets : 14.813221216202
- temps avec les quotes : 14.824532032013

Comme vous pouvez vous en rendre compte, dans les deux cas, les temps d'exécution sont sensiblements identiques.

Vous avez maintenant en main toutes les cartes pour choisir quel type de chaîne utiliser. Alors, apostrophe ou guillemet ? ;)