











Variables prédéfinies et variables externes

Sommaire

I. Variables prédéfinies Server	3
II. Variables externes de type GET, Passage de paramètres par l'URL	4
A. Méthode GET et formulaire.....	5
B. Remarque sur la limite en taille des URL	6
III. Variables externes de type POST	6

Crédits des illustrations:
© DR

Les repères de lecture

	Retour au chapitre		Définition		Objectif(s)		Espace Élèves		Vidéo / Audio
	Point important / À retenir		Remarque(s)		Pour aller plus loin		Normes et lois		Quiz

I. Variables prédéfinies Server

PHP fournit un grand nombre de variables prédéfinies qui sont prêtes à l'emploi et disponibles en permanence.

Ces variables fournissent par défaut des informations variées que nous pouvons utiliser directement dans nos scripts. Ces variables ont une syntaxe particulière.

Voyons par exemple les variables prédéfinies de type « server ».

Tableau n°1

<code>\$_SERVER['PHP_SELF']</code>	Cette variable contient le nom du fichier du script en cours d'exécution, par rapport à la racine web.
<code>\$_SERVER['SERVER_NAME']</code>	Cette variable contient le nom du serveur hôte qui exécute le script suivant.
<code>\$_SERVER['HTTP_USER_AGENT']</code>	Cette variable contient l'en-tête User_Agent (c'est à dire les informations relatives au navigateur).
<code>\$_SERVER['REMOTE_ADDR']</code>	Cette variable contient l'adresse IP du client qui demande la page courante.
<code>\$_SERVER['HTTP_REFERER']</code>	Cette variable contient indique quelle est la page de provenance de l'utilisateur.

Puisque ces variables prédéfinies sont des... variables, il est aisé d'en afficher les valeurs.

```
echo $SERVER['PHP_SELF'] ; // affiche : /test/variablespre.php qui est le
chemin du fichier PHP depuis c:\wamp\www
echo $SERVER['SERVER_NAME'] ; // affiche localhost, qui est le nom du serveur
echo $SERVER['HTTP_USER_AGENT'] ; // indique= quel est le « user_agent »,
autrement dit le navigateur : Mozilla/5.0 (Windows NT 6.1; WOW64; rv:15.0)
Gecko/20100101 Firefox/15.0.1. Le navigateur ici est firefox
echo $SERVER['REMOTE_ADDR'] ; // affiche 127.0.0.1, qui est l'adresse IP
```

Fig. 1

Comme on le voit, ces informations s'utilisent directement « en l'état ». Elles sont souvent utilisées dans les projets.

Il existe d'autres variables prédéfinies de type « server ».

Pour les connaître, il existe deux moyens :

- soit en effectuant une recherche 'variables prédéfinies' dans le manuel ;
- soit en effectuant un `print_r` car nous remarquons l'emploi des crochets qui indiquent vraisemblablement qu'il s'agit d'un tableau `$_SERVER`.

```
print_r($_SERVER);
```

Fig. 2

Ce `print_r` va retourner toutes les variables de type « server » qui sont disponibles dans le tableau `$_SERVER` et nous permettre d'afficher leur valeur comme suit :

```
Array
(
    [HTTP_HOST] => localhost
    [HTTP_USER_AGENT] => Mozilla/5.0 (Windows NT 6.1; etc...
    [HTTP_ACCEPT] => text/HTML,application/xhtml+xml,etc
    [HTTP_ACCEPT_LANGUAGE] => fr,fr-fr;q=0.8,en-us;q=0.5,en;q=0.3
    [HTTP_ACCEPT_ENCODING] => gzip, deflate
    [HTTP_CONNECTION] => keep-alive
    [HTTP_REFERER] => http://localhost/cours/variables.php
    [HTTP_CACHE_CONTROL] => max-age=0
    [PATH] => C:\Program Files\Common Files\etc...
    [SystemRoot] => C:\Windows
    [COMSPEC] => C:\Windows\system32\cmd.exe
    [PATHEXT] => .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE; etc...
    [WINDIR] => C:\Windows
    [SERVER_SIGNATURE] =>
    [SERVER_SOFTWARE] => Apache/2.2.21 (Win64) PHP/5.3.8
    [SERVER_NAME] => localhost
    [SERVER_ADDR] => 127.0.0.1
    [SERVER_PORT] => 80
    [REMOTE_ADDR] => 127.0.0.1
    [DOCUMENT_ROOT] => C:/wamp/www/
    [SERVER_ADMIN] => admin@localhost
    [SCRIPT_FILENAME] => C:/wamp/www/cours/variables.php
    [REMOTE_PORT] => 53235
    [GATEWAY_INTERFACE] => CGI/1.1
    [SERVER_PROTOCOL] => HTTP/1.1
    [REQUEST_METHOD] => GET
    [QUERY_STRING] =>
    [REQUEST_URI] => /cours/variables.php
    [SCRIPT_NAME] => /cours/variables.php
    [PHP_SELF] => /cours/variables.php
    [REQUEST_TIME] => 1349788436
)
```

Fig.3

Toutes ces variables `$_SERVER` prédéfinies par PHP sont prêtes à l'emploi en fonction des besoins.

II. Variables externes de type GET, Passage de paramètres par l'URL

Les URL permettent de consulter les pages dans un navigateur.

Exemple

L'URL `http://localhost/test.php` permet de consulter la page `test.php` qui est hébergée sur le serveur `localhost` (rappel : il s'agit du serveur web local géré par Wamp).

L'URL permet également de véhiculer des informations via la **méthode GET**. Cette méthode GET permet de récupérer des variables passées dans l'URL (c'est à dire présentes dans l'URL).

Il est possible par exemple d'avoir une URL du type **`http://localhost/test.php?couleur=bleu`**

Nous constatons qu'après le nom de la page **`test.php`**, un **point d'interrogation** est écrit, suivi du mot `couleur`, puis du signe égal puis du mot `bleu`. Ici `couleur` est le nom de la variable passée dans l'URL et `bleu` est la valeur qu'elle contient. Le point d'interrogation sert à indiquer que nous avons une variable contenue dans l'URL.

Pour accéder à cette valeur transmise par la méthode GET, nous utilisons la **variable externe de type GET** dont la syntaxe est la suivante :

```
// récupération de la variable couleur dans l'URL.  
echo $_GET['couleur']; // affiche bleu
```

Fig.4

Nous pourrions avoir plusieurs valeurs dans l'URL. Dans ce cas, les variables seraient séparées par le signe « & » (ce signe est parfois dénommé « et commercial ») comme le montre cet exemple d'URL : `http://localhost/test.php ?couleur=bleu&modele=607`

Dans ce cas deux variables sont passées dans l'URL.

```
// récupération de la variable couleur et de la variable modele dans l'URL.  
echo $_GET['couleur']; // affiche bleu  
echo $_GET['modele']; // affiche 607
```

Fig.5

Nous remarquons que des crochets sont utilisés et qu'ils indiquent vraisemblablement qu'il s'agit d'un tableau `$_GET`. Nous pouvons le vérifier en effectuant un `print_r` :

```
print_r($_GET);
```

Fig.6

Ce `print_r` va nous retourner l'ensemble des variables externes de type GET comme ceci :

```
Array  
(  
    [couleur] => bleu  
    [modele] => 607  
)
```

Fig.7

A. Méthode GET et formulaire

Il est possible de produire une URL avec un formulaire HTML lorsque la méthode GET est stipulée dans l'attribut *method* de l'élément `<form>` ou lorsque cet attribut n'est pas rempli (il s'agit dans ce cas du fonctionnement par défaut).

Exemple de formulaire fonctionnant en méthode GET :

```
<form action="cible.php" method="get">  
<input type="text" name="prenom">  
<input type="submit">  
</form>
```

Fig.8

Voici le rendu dans le navigateur de ce code HTML :



Fig.9

Si nous saisissons le prénom « Claude » dans le champ de saisie, et que nous **soumettons** (validons) le formulaire en cliquant sur le bouton de soumission (ou bouton de validation) « Envoyer », le formulaire nous redirige vers la page *cible.php* (qui est indiquée dans l'attribut *action*) avec l'URL suivante :

```
http://localhost/essai/cible.php?prenom=claudio
```

Fig. 10

Remarquons que la variable externe sera nommée *prenom* de la **même façon** que le champ *prenom* du formulaire. La valeur de cette variable sera naturellement celle que nous avons saisie.

Ajoutons maintenant un nouveau champ email au formulaire :

```
<form action="cible.php" method="get">
<input type="text" name="prenom">
<input type="text" name="email">
<input type="submit">
</form>
```

Fig. 11

L'URL cible produite suite à la soumission du formulaire sera la suivante :

```
http://localhost/essai/cible.php?prenom=claudio&email=test@test.com
```

Fig. 12

Nous constatons que le formulaire gère automatiquement l'ajout des « & » pour séparer les variables. Pour récupérer ces variables, nous utilisons la syntaxe habituelle :

```
echo $_GET['prenom'] ; //affiche Claude
echo $_GET['email'] ; //affiche test@test.com
```

Fig. 13

B. Remarque sur la limite en taille des URL

Notons que le paramétrage des serveurs fait que souvent les URL sont limitées en taille (de 255 caractères à... plusieurs milliers). Cela ne sera pas souvent une contrainte mais il s'agit tout de même d'une limite à l'emploi de la méthode GET lorsque nous aurons besoin de transférer des données de taille importante.

III. Variables externes de type POST

Les variables externes de type POST fonctionnent de la même manière que les variables externes de type GET à l'exception notable près que ces variables ne sont pas visibles dans l'URL.

Les variables externes de type POST sont transmises par la méthode POST qui est supportée par le protocole HTTP. Le protocole HTTP est ce qui permet au serveur d'envoyer les informations lors de la requête client-serveur.

Le fait que les informations transmises soient cachées car non visibles peut être, selon les cas, un avantage de la méthode POST par rapport à la méthode GET. Le deuxième avantage est que la méthode POST ne limite pas en taille les informations transmises.

Puisque les variables externes de type POST ne sont pas visibles dans l'URL, elles ne peuvent être produites ni en saisissant une URL dans la barre d'adresse du navigateur ni en cliquant sur un lien.

La seule manière de produire une variable externe de type POST est d'utiliser un formulaire HTML.

Reprenons le formulaire simple avec les deux champs *prenom* et *email*, mais précisons cette fois dans l'attribut *method* que nous utilisons la méthode POST.

```
<form action="cible.php" method="post">
<input type="text" name="prenom">
<input type="text" name="email">
<input type="submit">
</form>
```

Fig. 14

Lorsque nous validons notre saisie, le formulaire va nous rediriger vers la page cible.php et l'URL ne contiendra pas de valeurs :

```
http://localhost/essai/cible.php
```

Fig. 15

En revanche les valeurs saisies seront bien postées et accessibles par PHP car elles sont stockées sur le serveur.

Pour les récupérer, nous utilisons la variable externe de type POST dont la syntaxe est : `$_POST`

```
echo $_POST['prenom'] ; //affiche la saisie utilisateur (Claude)
echo $_POST['email'] ; //affiche la saisie utilisateur (test@test.com)
```

Fig. 16

La différence entre les syntaxe GET et POST tient seulement dans le changement de GET en POST, le préfixe `$_` ne change pas.

De la même manière que pour GET, les informations contenues dans les variables de type POST forment un tableau de données et nous pouvons effectuer *un print_r* comme ceci :

```
print_r($_POST);
```

Fig. 17

Ce *print_r* va nous retourner l'ensemble des variables de type POST qui ont été transmises lors de la soumission du formulaire :

```
Array
(
    [prenom] => Claude
    [email] => test@test.com
)
```

Fig. 18

