Programmation en PHP
Cyril Beaussier

Débuter avec les sessions

Version 1.0a – Octobre 2004

COPYRIGHT ET DROIT DE REPRODUCTION

Si vous souhaitez des améliorations, je suis évidemment ouvert à toute suggestion. Il en est de même si vous constatez une erreur (nul n'est parfait). Pour cela, il suffit de m'écrire avec pour sujet « *Programmation en PHP / Débuter avec les sessions* » dans la rubrique « *Contact* » de mon site principal :

www.beaussier.com

En revanche, je n'assure aucune aide, ni support sur des questions de programmation ou de compréhension de ce manuel. Je vous invite à vous reporter sur les excellents forums de Developpez.com (section *PHP*) où je traine régulièrement.

Je ne suis lié avec aucun éditeur ou constructeur informatique.

Ce support a été réalisé avec la suite bureautique libre *Open Office* 1.1 (disponible gratuitement sur http://fr.openoffice.org) qui permet d'exporter nativement en PDF.

Avertissement complémentaire :

Les éléments (données ou formulaires) éventuellement inclus dans ce support vous sont fournis à titre d'exemple uniquement. Leur utilisation peut avoir, dans certains cas, des conséquences matériels et juridiques importantes qui peuvent varier selon le sujet dont ils traitent. Il est recommandé d'être assisté par une personne compétente en informatique ou de consulter un conseiller juridique ou financier avant de les utiliser ou de les adapter à votre activité.

Sommaire

1. Introduction

Vous allez vous dire « *encore un énième support sur les sessions en PHP* ». Cependant, lorsque j'ai commencé à m'atteler à cet aspect important de la programmation sur ce langage, je n'ai pas trouvé de didacticiel satisfaisant certaines de mes interrogations. J'ai donc décidé d'écrire ce support histoire d'apporter mon point de vue.

La gestion des sessions en PHP a été ajoutée à partir de la version 4. Je pars du principe que vous utilisez au minimum PHP dans sa version **4.1.0** (qui est d'ailleurs déjà ancienne). Ceci est notamment valable pour la syntaxe des fonctions et surtout des variables **superglobales**.

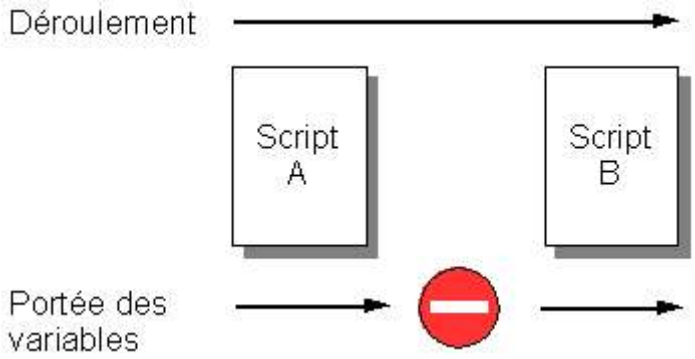
Si vous développez avec une version antérieure, je vous conseille de vous reporter sur la documentation officielle pour la correspondance des fonctions.

Pour ce support, j'ai télédéchargé le célèbre **EasyPHP** 1.7 (disponible gratuitement sur www.easyphp.org). Je ne détaille pas la procédure d'installation et vous renvoie sur leur site pour ces détails.

2. Concept

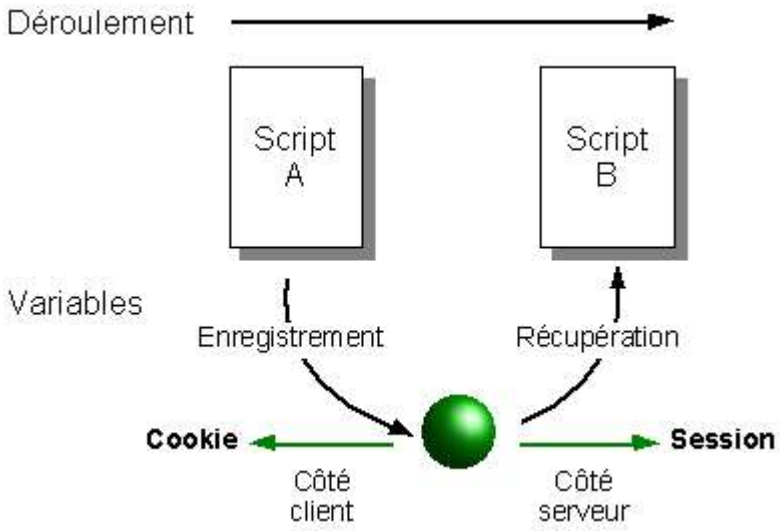
Lorsque vous créez un site web, vous avez rapidement besoin de stocker et d'afficher des informations sur vos utilisateurs pour les distinguer ou leur attribuer des droits d'utilisation.

Le problème principal des langages comme PHP, c'est que les variables n'ont qu'une durée de vie limitée à celle du script qui les appelle.



Pour conserver des données de page en page, il faut alors les passer par la méthode **GET** ou **POST** (voir chapitre 3. Restrictions). Ce qui peut être contraignant si l'on a beaucoup de données à conserver ou que l'on ne veuille pas les faire apparaître pour le client.

On peut heureusement avoir recours à un mécanisme de stockage de ces variables et pouvoir les récupérer par la suite.



Il existe deux moyens de stockage d'informations. Ils sont tous les deux sous la forme de fichier enregistré sur le disque :

- Le **cookie** qui sera côté client.
- La **session** qui sera côté serveur.

Il suffit d'enregistrer les variables dans le fichier depuis le script A pour les lire ensuite dans le script B.

Le problème majeur du cookie c'est que le client a le pouvoir de le refuser. Même si c'est devenu une exception, votre application risque donc de ne pas pouvoir fonctionner.

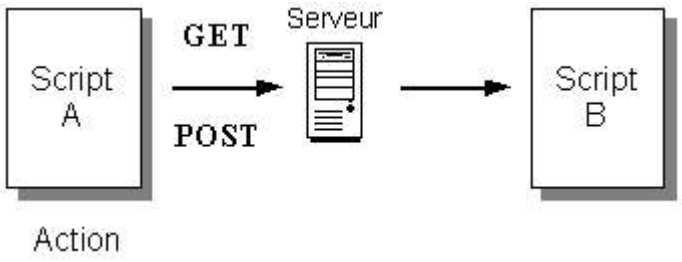
Il y a aussi des risques plus graves quant à la sécurité. L'usurpation d'identité, car ce fichier peut être recopié facilement sur un autre ordinateur. La manipulation, car le cookie n'est qu'un simple fichier texte dont il est alors aisé de changer les informations.

La session n'aura donc pas cet inconvénient puisque tout est géré sur le serveur de l'application auquel le client n'a pas accès directement.

Autre avantage, vous n'avez plus à vous soucier du passage des informations de page en page (voir chapitre 3. Restrictions). Vos URL gagnent ainsi en clarté et vos formulaires HTML sont allégés des champs <INPUT HIDDEN> avec des données qui sont alors stockées dans la session.

3. Restrictions

Dans tous les cas, il vous faudra garder à l'esprit que le client reste **maître** du déroulement de l'application. Seule une action de sa part permet l'envoi des informations au serveur pour la génération de la page suivante.



Cette action peut être sous la forme :

- d'un **GET** avec le passage de l'information dans l'URL (barre d'adresse) du navigateur ;
- d'un **POST** avec un postage d'information depuis un formulaire HTML.

La session (tout comme le cookie d'ailleurs) ne permet le stockage que de variables de type *primitif*. Ce sont les chaînes de caractères, les nombres (entier, réel, flottant) mais également les tableaux.

Autre restriction, la durée de vie d'une session est également définie au niveau serveur. Sa valeur peut être consultée depuis un phpinfo() à la ligne session.cache_expire et modifiée dans le fichier de configuration (php.ini).

Par défaut sous **EasyPHP**, elle est paramétrée à 180 minutes soit 3 heures. Cela veut dire que si votre client laisse son navigateur ouvert sur votre application pendant 3 heures et 1 seconde sans exécuter d'action (GET ou POST), sa session sera détruite automatiquement par le serveur.

Il vous appartient de paramétrer cette variable en fonction de vos besoins.

Certains hébergeurs ne permettent pas de modifier cette durée, ce peut être un problème important lors du passage en production.

4. Sécurité

Le mécanisme des sessions n'est pas la panacée universelle pour la sécurité de votre application. Une session en PHP repose sur un identifiant (une suite de chiffres et de lettres) qui peut être volée par un autre utilisateur. Cela peut arriver si l'identifiant apparaît en clair dans la barre d'adresse du navigateur et qu'un utilisateur envoie par mégarde un courriel contenant l'URL à un de ces correspondants ou l'enregistre dans ces favoris.

Si cela se produit, il faut alors vous demander quelles seront les conséquences si un utilisateur malveillant accède aux informations stockées. Cela peut s'avérer grave surtout si un mot de passe ou des coordonnées bancaires y sont enregistrés.

Ce vol peut cependant être rendu plus difficile en couplant la session avec un cookie ou en utilisant l'adresse IP de l'utilisateur. A vous de gérer cet aspect et programmer votre application pour un contrôle plus étroit et réagir efficacement dans ces cas-là.

5. Fonctions

Pour utiliser les sessions, PHP vous propose un éventail de commandes décrites dans le tableau ci-dessous :

session_cache_expire	Obtenir la configuration du cache expire
session_cache_limiter	Lecture/écriture pour le limiteur de cache
session_decode	Décode les données de session
session_destroy	Détruit une session
session_encode	Encode les données de session
session_get_cookie_params	Lit la configuration du cookie de session
session_id	Lecture/écriture pour l'identifiant de la session courante
session_is_registered	Vérifie si une variable existe dans la session
session_module_name	Lecture/écriture pour le module de session courant
session_name	Lecture/écriture pour le nom de la session
session_readonly	Initialise une session en mode lecture
session_register	Enregistrement d'une variable dans une session
session_save_path	Lecture/écriture pour le chemin de sauvegarde des sessions
session_set_cookie_params	Modifie les paramètres du cookie de session
session_set_save_handler	Configure les fonctions de stockage de sessions
session_start	Initialise une session
session_unregister	Supprime une variable de la session
session_unset	Détruit toutes les variables de session
session_write_close	Ecriture de données et fermeture de la session

Les commandes marquées en rouge sont déclarées comme **obsolètes** car elles posent problème dans les environnements où la directive **registrer_globals** est désactivée. L'utilisation de celles-ci risque donc de provoquer des dysfonctionnements dans votre application.

Vous devez donc **impérativement** utilisez la superglobale **$_SESSION** pour l'enregistrement ou l'effacement des données. C'est ce que nous allons étudier dans ce support.

6. Mécanisme

Le mécanisme des sessions est vraiment extrêmement simple. Tout ce que vous devez faire, c'est systématiquement demander au compilateur PHP de démarrer une session pour le client.

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

Cette commande doit ainsi toujours figurer sur chacun de vos scripts. Elle est obligatoire si la directive session.auto_start est à *Off*. Si vous passez celle-ci à *On*, Le démarrage de session sera automatique.

Le compilateur PHP va alors créer dans le répertoire de sauvegarde des sessions, un fichier dont le nom commence par sess_ et se termine par un identifiant généré de manière aléatoire.

Le répertoire de sauvegarde des sessions peut être consulté par un phpinfo() à la ligne session.save_path et modifié par la commande session_save_path().

L'identifiant de session peut être affiché par la commande session_id(). Vous pouvez également gérer vous-même ce nom de session en utilisant session_name() avant le démarrage de la session.

La session est perdue définitivement pour l'utilisateur lorsque :

- Il n'a exécuté aucune action (POST ou GET) au delà de la durée de vie paramétrée par session.cache_expire.
- Il ferme son navigateur.
- La commande session_destroy est appelée.

7. Enregistrement

Prenons un premier exemple pour voir comment s'enregistre une variable dans une session avec un formulaire simple pour sauvegarder le nom du client qui s'est connecté.

Créez d'abord un répertoire *Test* à la racine de votre site web local. Si vous êtes avec **EasyPHP**, ce sera : [`C:\Program Files\EasyPHP1-7\www`].

Avec votre éditeur favori, saisissez votre premier script :

Page1.php

```
<?php
session_start();
?>
<html>
<body>
<form method="POST" action="page2.php">
Entrez votre  nom : <input type="TEXT" name="nom">
<input type="SUBMIT" value="OK">
</form>
</body>
</html>
```

Vous avez donc le démarrage de la session puis un simple formulaire HTML qui va poster au script *page2.php* le contenu de la variable *nom*.

Remarque :

La commande session_start ne sert pas vraiment car elle n'est utile qu'à partir du script suivant. Néanmoins, elle permet d'être initialisée au lancement de l'application par le client.

Testez votre script avec votre navigateur :

Entrez votre nom : OK

Continuons avec le second script...

Page2.php

```
<?php
session_start();
$nom = $_POST['nom'];
$_SESSION['nom'] = $nom;
?>
<html>
<body>
Bienvenue sur ce site <b><?php echo $nom; ?></b>.<br />
Regardons ce qui se passe sur la
<a href="page3.php">page</a> suivante.<br />
</body>
</html>
```

Dans ce script, nous avons donc le démarrage de la session puis l'enregistrement dans une variable *$nom* de la valeur postée par le formulaire. Enfin, nous enregistrons *$nom* dans une variable de session.

Bien sûr, nous aurions pu faire également :

```
$_SESSION['nom'] = $_POST['nom'];
```

Nous affichons ensuite la variable puis un lien vers la page suivante, ce qui va nous donner à l'affichage :

Bienvenue sur ce site **BiD0uille**.
Regardons ce qui se passe sur la page suivante.

Passons au script suivant...

Page3.php

```
<?php
session_start();
$nom = $_SESSION['nom'];
?>
<html>
<body>
Vous êtes toujours parmi nous
<b><?php echo $nom; ?></b>.<br />
</body>
</html>
```

Dans ce script nous avons toujours le démarrage de session avec cette fois, la récupération de la variable *nom* depuis le tableau superglobal de session. Nous procédons enfin à l'affichage de la variable dans une phrase :

Vous êtes toujours parmi nous **BiD0uille**.

Comme vous le voyez, nous n'avons pas eu besoin de passer le nom en URL (GET) ou par formulaire (POST). L'utilisation de la session permet d'avoir tout le temps les variables stockées sous la main.

8. Effacement

Nous avons vu dans le chapitre précédent que l'enregistrement de variables était vraiment très facile. Pour effacer une variable de session, c'est tout aussi simple.

Reprenons notre script et ajoutons quelques lignes :

Page3.php

```
<?php
session_start();
$nom = $_SESSION['nom'];
?>
<html>
<body>
Vous êtes toujours parmi nous
<b><?php echo $nom; ?></b>.<br />
Effacement de votre nom en cliquant
<a href="page4.php">ici</a>.<br />
</body>
</html>
```

Ce qui nous donne dans le navigateur :

Vous êtes toujours parmi nous **BiD0uille**.
Effacement de votre nom en cliquant ici.

Continuons en créant notre quatrième script :

L'effacement d'une variable de session se fait de la même manière qu'avec une variable classique. Vous utilisez la commande unset() suivie du tableau superglobal et de l'index contenant votre variable :

Page4.php

```
<?php
session_start();
?>
<html>
<body>
<?php
unset ($_SESSION['nom']);
if ( isset ($_SESSION['nom'])) {
      $resultat = "La suppression a échouée ."; }
else {
      $resultat = "Votre nom a été effacé."; }
echo $resultat;
?>
<br />
Repartons en <a href="page3.php">arrière</a>.<br />
</body>
</html>
```

On peut éventuellement tester par isset() que la variable de session a bien été effacée.

L'exécution du script affichera donc :

Votre nom a été effacé.
Repartons en arrière.

9. Test d'existence

Si nous cliquons sur le lien de *Page4.php* et que nous affichons à nouveau le script *Page3.php*, nous obtenons une notification d'erreur de la part du compilateur :

Notice: Undefined index: nom in
c:\program files\easyphp1-7\www\test\page3.php on line **3**
Vous êtes toujours parmi nous .
Effacement de votre nom en cliquant ici.

Remarque :

Le message nous informe que l'index appelé dans le tableau des sessions n'existe pas. Cela arrive très souvent aux novices.

Il est donc important de savoir si les variables de session que nous appelons existent bien avant de les récupérer.

Nous avons d'ailleurs déjà vu dans le script précédent comment tester l'existence d'une variable de session. Nous allons donc à nouveau utiliser la commande isset().

Reprenons notre script et modifions quelques lignes...

Page3.php

```
<?php
session_start();
if ( isset ($_SESSION['nom']) ) {
      $nom = $_SESSION['nom']; }
else {
      $nom = "mais votre nom a été effacé"; }
?>
<html>
<body>
Vous êtes toujours parmi nous
<b><?php echo $nom; ?></b>.<br />
Effacement de votre nom en cliquant
<a href="page4.php">ici</a>.<br />
</body>
</html>
```

Voilà, vous pouvez tester à nouveau et constatez que maintenant l'affichage est à nouveau correct :

Vous êtes toujours parmi nous **mais votre nom a été effacé**.
Effacement de votre nom en cliquant ici.

10. Effacement total

Dans notre exemple, nous n'avons qu'une seule valeur stockée dans notre session. Cependant, il peut arriver de vouloir réinitialiser toutes les variables sans pour autant vouloir changer de session.

Pour effacer l'ensemble des valeurs d'une session, vous avez deux possibilités :

- Utiliser la commande session_unset() qui ne prend aucun paramètre et ne retourne aucune valeur.

```
session_unset();
```

- Vider le tableau superglobal des sessions.

```
$_SESSION = array();
```

Remarque :

Il est recommandé par la Communauté PHP d'utiliser la seconde possibilité avec l'effacement du tableau superglobal.

11. Destruction

Comme nous l'avons vu au chapitre 6. Mécanisme, la session s'arrête si :

- La durée de vie maximum est atteinte
- L'utilisateur n'a fait aucune action dans l'application
- L'utilisateur a fermé son navigateur.

On peut donc se demander s'il est nécessaire de détruire une session.

Vous pouvez avoir besoin de détruire une session si votre application gère les accès des utilisateurs. Vous aurez bien sûr un formulaire d'identification (login et mot de passe) pour la connexion. Il est alors utile d'avoir également un lien pour la déconnexion et permettre ainsi de changer d'utilisateur.

Reprenons une dernière fois notre script...

Page3.php

```
<?php
session_start();

if ( isset ($_SESSION['nom'])) {
      $nom = $_SESSION['nom']; }
else {
      $nom = "mais votre nom a été effacé"; }
?>
<html>
<body>
Vous êtes toujours parmi nous
<b><?php echo $nom; ?></b>.<br />
Effacement de votre nom en cliquant
<a href="page4.php">ici</a>.<br />
Effacement de votre session en cliquant
<a href="page5.php">ici</a>.<br />
</body>
</html>
```

On ajoute un lien vers le script *Page5.php* pour permettre la destruction de la session. Ce qui nous donne à l'affichage :

Vous êtes toujours parmi nous **mais votre nom a été effacé**.
Effacement de votre nom en cliquant ici.
Effacement de votre session en cliquant ici.

Terminons par le script de destruction de session...

Page5.php

```
<?php
session_start();
session_destroy();
?>
<html>
<body>
Votre session a été détruite.
</body>
</html>
```

Il n'y a rien de plus à faire. La commande session_destroy() ne prend aucun paramètre.

Une fois exécuté, votre navigateur affichera

> Votre session a été détruite.

12. Conclusion

Voilà, nous venons de couvrir les bases pour l'utilisation des sessions en PHP. Vous savez désormais comment démarrer une session, y enregistrer une variable, l'effacer et enfin détruire la session elle-même.

Je vous invite également sur mon site principal où vous retrouverez d'autres supports. Reportez-vous pour cela à l'adresse suivante : www.beaussier.com/?pg=doc