

Petite FAQ sur les Sessions PHP

Dernière mise à jour : le 30 avril 2003.

Par Hugo ETIEVANT

Sommaire

1. Préambule
2. Où trouver de la documentation sur les sessions ?
3. Comment conserver des variables de page en page ?
4. Pourquoi utiliser les sessions ?
5. Comment fonctionne une session ?
6. Comment démarrer une session ?
7. Comment sauvegarder une variable dans une session ?
8. Comment supprimer une variable d'une session ?
9. Comment savoir si une variable appartient déjà à la session ?
10. Quelle est la durée de vie d'une session ?
11. Comment fermer une session ?
 - a. Fermeture conservatrice
 - b. Fermeture destructive
12. Quelles sont les méthodes PHP associées aux sessions ?
13. Quelles sont les erreurs possibles ?
 - a. Répertoire de session inaccessible
 - b. PHP n'est pas autorisé à utiliser les sessions
 - c. Avoir déjà écrit dans la page
 - d. Tentative d'envoi d'entêtes
 - e. Défaut de déclaration de classe
14. Qu'est-ce que sont les entêtes ?
 - a. Exemples d'entêtes
 - b. Envoyer des entêtes
15. Exemple qui marche
16. Comment configurer php.ini
17. Cas particulier de l'hébergeur FREE.FR
18. Comment faire cohabiter sur le même serveur deux applications utilisant les sessions ?

Préambule

Cette petite FAQ n'a pas la prétention d'expliquer de façon exhaustive le système de session de PHP. Elle peut être améliorée. Pour toute suggestion : écrire à l'auteur.

Les informations que vous y trouverez s'inspirent largement des messages postés sur le forum **Developpez.com > Forum > PHP**

Se rendre sur le forum PHP.

[haut]

Où trouver de la documentation sur les sessions ?

- <http://www.php.net/manual/fr/ref.session.php>
- http://cyberzoide.developpez.com/php4/php4_mysql.pdf (Chapitre Sessions)

[haut]

Comment conserver des variables de page en page ?

- En stockant leurs valeurs dans une base de données.
- En les enregistrant dans un fichier sur le serveur (mais oblige à donner des droits d'écriture aux visiteurs).
- Utiliser des cookies (malheureusement certains navigateurs ne les acceptent pas).
- En les envoyant dans l'URL (mais seront visibles aux visiteurs dans la barre d'adresse et leur impose de toujours utiliser les liens hypertextes).
- En les passant par un formulaire (mais impose aux visiteurs de cliquer sur un bouton d'envoi).
- En utilisant une session.

[haut]

Pourquoi utiliser les sessions ?

- Pour conserver de page en page les valeurs de certaines variables.
- Pour pister le parcours du visiteur.
- Pour effectuer des statistiques fines en termes de visiteurs réels et pas en hits (nombre d'appel d'un fichier).

[haut]

Comment fonctionne une session ?

Les session permettent tout du long de la visite d'un internaute sur le site, de conserver des informations de façon transparentes.

Cela est sans ralentissement de performances car le client ne stocke sous forme de cookies que l'ID de session (généralisé aléatoirement), le serveur stockant sur disque le contenu des variables dans le répertoire défini par `session.save_path`.

Les sessions sont activées manuellement par la commande **session_start()** ou automatiquement si `session.auto_start` est à 1 ou encore implicitement par la commande **session_register()**.

Le serveur attribut à chaque visiteur un identifiant unique qui est soit envoyé au client sous forme de cookie (par défaut) soit passé de façon systématique dans l'URL.

[haut]

Comment démarrer une session ?

Il existe trois manière de démarrer une session :

- Automatique si `session.auto_start` est à 1.
- Manuellement avec la commande **session_start()**.
- Implicitement par la commande **session_register()**.

Syntaxe : `boolean session_start ( void )` Crée une session ou continue la session courante, en fonction de l'identifiant de session passé par l'URL (méthode GET) ou par un cookie. Exemple :

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

Comment sauvegarder une variable dans une session ?

Par l'usage de la commande **session_register()** dont voici la syntaxe :

```
boolean session_register ( mixed name [, mixed ...])
```

Elle prends en paramètre une chaîne de caractères contenant le nom de la variable à sauvegarder ou bien un tableau de chaînes de caractères ou encore un tableau de tableaux...

Exemple avec une variable chaîne de caractères :

```
<?php
session_start() ;
$foobar = "Hello Word !";
session_register("foobar");
...
?>
```

Autre exemple avec le tableau superglobal `$_SESSION`:

```
<?php
session_start() ;
$_SESSION['foobar'] = "Hello Word !";
...
?>
```

Toute modification ultérieure dans le script des variables de session (avant la fermeture de la session) sera automatiquement répercutée autant dans la session que dans l'espace mémoire des données du script en exécution.

Tous les types de variable sont acceptés par **session_register()** : chaînes, nombres, tableaux, objets (dont la classe doit être incluse avant le démarrage de la session).

Exemple avec une variable objet :

```
<?php
require("../common/visitor.class.php");
session_start() ;
$myVisitor = new Visitor();
session_register("myVisitor");
...
?>
```

Comment savoir si une variable appartient déjà a la session ?

La commande **session_is_registered** renvoie VRAI si la variable dont l'identificateur est passé en paramètre a déjà été enregistré parmi les variables de session.

Syntaxe :

```
boolean session_is_registered ( string name)
```

Exemple :

```
<?php
session_start() ;
$foobar = "toto";
if(!session_is_registered("foobar")) {
    session_register("foobar");
}
...
?>
```

Comment supprimer une variable d'une session ?

La commande **session_unregister()** supprime une variable dans la session courante.

Elle a pour syntaxe :

```
boolean session_unregister (string name)
```

Elle retourne TRUE si success, FALSE sinon.

Il est aussi possible de purger toutes les variables de la session avec **session_unset()**.

Si vous utilisez le tableau superglobal **\$_SESSION**, il suffit alors d'utiliser **unset()** :

```
unset($_SESSION["mavARIABLE"])
```

[haut]

Quelle est la durée de vie d'une session ?

Dès que l'on ferme le navigateur la session est détruite. Sauf à configurer le fichier **php.ini** avec **session.lifetime** qui fixe la durée de vie, en secondes, du cookie envoyé au client. La valeur 0 signifie "jusqu'à ce que le client soit fermé". Par défaut à 0. Donc si on le laisse à zéro la session dure jusqu'à la fermeture du navigateur, pour laisser les données durant 30 minutes, il faut remplacer 0 par 1800 (= 30 minutes * 60 secondes dans une minute).

```
session.lifetime = 0
```

[haut]

Comment fermer une session ?

Fermeture conservatrice :

La commande **session_write_close()** écrit les valeurs des variables de session sur le serveur et ferme la session.

Fermeture destructive :

La commande **session_destroy()** détruit toutes les données enregistrées d'une session. Cette dernière commande est la plus utilisée car n'impose aucune sauvegarde au serveur. Retourne TRUE en cas de succès, et FALSE sinon.

[haut]

Quelles sont les méthodes PHP associées aux sessions ?

session_cache_expire() -- Retourne la date d'expiration du cache de la session

session_cache_limiter() -- Lit et/ou modifie le limiteur de cache

session_decode() -- Décode les données de session à partir d'une chaîne

session_destroy() -- Détruit toutes les données enregistrées d'une session

session_encode() -- Encode les données de session dans une chaîne

session_get_cookie_params() -- Lit les paramètres du cookie de session

session_id() -- Affecte et/ou retourne l'identifiant de session courante

session_is_registered() -- Indique si une variable a été enregistrée dans la session ou pas

session_module_name() -- Affecte et/ou retourne le module courant de session courante

session_name() -- Affecte et/ou retourne le nom de la session courante

session_readonly() -- Lit les variables de session sans verrouiller les données

session_register() -- Enregistre une variable dans la session courante

session_save_path() -- Affecte et/ou retourne le chemin de sauvegarde de la session courante

session_set_cookie_params() -- Modifie les paramètres du cookie de session

session_set_save_handler() -- Définit les fonctions utilisateurs de stockage des sessions

session_start() -- Initialise les données de session
session_unregister() -- Supprime une variable dans la session courante
session_unset() -- Détruit toutes les variables de session
session_write_close() -- Ecrit les données de sessions et termine la session

[haut]

Quelles sont les erreurs possibles ?

Répertoire de session inaccessible

Warning: open(/tmp\sess_3c80883ca4e755aa72803b05bce40c12, O_RDWR) failed: m (2) in c:\phpdev\www\bp\header.php on line 2

ou encore

PHP Warning: Unknown(): open(/tmp\sess_3c80883ca4e755aa72803b05bce40c12, O_RDWR) failed: No such file or directory (2) in Unknown on line 0
PHP Warning: Unknown(): Failed to write session data (files). Please verify that the current setting of session.save_path is correct (/tmp) in Unknown on line 0

Cette erreur est due à l'absence du répertoire de sauvegarde (ici /tmp) des sessions ou bien au manque du droit d'écriture dans ce répertoire pour les visiteurs (utilisateur nobody, www-data ou autre... sous Apache).

Le répertoire de sauvegarde est défini dans le **php.ini** : `session.save_path = /tmp`

Il faut donc:

1. Créer un répertoire
2. Lui donner les droits d'écriture pour tous
3. En spécifier le chemin dans le **php.ini**

PHP n'est pas autorisé à utiliser les sessions

Il faut s'assurer que le PHP est bien autorisé à créer des sessions. C'est juste un paramètre à activer. Faire un **phpinfo()** pour voir ces paramètres. La commande **phpinfo()** se contente d'afficher dans le navigateur le contenu du fichier de configuration **php.ini**.

Avoir déjà écrit dans la page

Warning: Cannot send session cookie - headers already sent by (output started at /home/SiteWeb/SiteAnalyse/index.php:3) in /home/SiteWeb/SiteAnalyse/index.php on line 6

Cette erreur survient lorsqu'on tente d'ouvrir une session après avoir déjà écrit dans le document, ce qui interdit, bien sûr.

Tentative d'envoi d'entêtes

Warning: Cannot add header information - headers already sent by (output started at /home/SiteWeb/SiteAnalyse/index.php:3) in /home/SiteWeb/SiteAnalyse/index.php on line 25

Cette erreur survient lorsqu'on tente d'envoyer des entêtes grâce à la fonction **header()** après avoir écrit dans la page.

On ne peut pas commencer une session après que le serveur ait envoyé au client les entêtes HTTP/1.0 (ou supérieures) de la page.

Ainsi, la commande **session_start()** doit impérativement être exécutée avant tout envoi par le serveur d'entêtes HTTP au navigateur.

L'identifiant de session étant envoyées sous forme de cookies au client, ce dernier doit être envoyé avant que la page ne s'affiche car l'affichage force l'envoi d'entêtes.

Tout contenu texte placé avant **session_start()** (même un saut de ligne) provoque un affichage et donc l'envoi d'entêtes qui doivent précéder contenu de la page.

Ce qu'il ne faut pas faire :

```
<html>
<body>
<?php session_start();
...
```

ceci non plus :

```
<?php echo "<html>";
...
session_start();
```

Car cela provoque l'envoi d'entêtes au navigateur. Donc ces deux essais sont erronés. Il faut faire le **session_start()** avant toute chose !!!

Même un simple saut de ligne dans le script avant **session_start()** provoque cette erreur.

Défaut de déclaration de classe

Fatal error: The script tried to execute a method or access a property of an incomplete object. Please ensure that the class definition utilisateur of the object you are trying to operate on was loaded _before_ the session was started in /home/SiteWeb/SiteAnalyse/test.php on line 12

Lorsqu'une variable que l'on veut enregistrer dans une session est un objet, PHP doit pouvoir en connaître la description, il faut donc déclarer les classes avant de faire un **session_start()**.

[haut]

Qu'est-ce que sont les entêtes ?

Le rôle des entêtes est d'échanger des méta-informations (informations à propos des informations échangées que sont les pages html générées ou non dynamiquement à partir de PHP) entre le serveur et le client.

Exemples d'entêtes

Server: Apache/1.3.9 (Unix) Debian/GNU qui renseigne le client sur la nature du serveur distant

Last-Modified: Sun, 07 Apr 2002 14:30:30 GMT qui donne la date de dernière modification du document

Envoyer des entêtes

La commande **header()** du php permet l'envoi d'entêtes personnalisées.
Par exemple :

```
header("Location: home2.php3")
```

pour rediriger le navigateur sur la page "home2.php3"

Les entêtes peuvent servir à la redirection, à l'authentification, à l'envoi d'images au navigateur...

[haut]

Exemple typique qui marche

```

<?php

/* démarrage session */
session_start();

/* si la variables $NOM
   n'existe pas alors : */
if(!isset($NOM)) {
    $NOM = "Durand";
    /* sauvegarde de la variable $NOM
       afin qu'elle soit connue dans
       les autres pages */
    session_register("NOM");
echo "init";
} else {
echo "pas d'init";
}
echo $NOM
echo "<a href=$PHP_SELF>recharger</a>"

?>

```

Au premier chargement de cette page, la variable \$NOM n'existera pas, on va donc la créer. Aux autres chargements, elle sera déjà présente (grâce à la session), on ne fera donc pas d'initialisation...

[haut]

Comment configurer php.ini

Ci-après les options de configuration des sessions du fichier **php.ini**

`session.save_handler`

définit les noms des fonctions qui seront utilisées pour enregistrer et retrouver les données associées à une session. Par défaut, les sessions sont enregistrées dans des fichiers. Mais on pourrait les enregistrer dans une base de données ; il faudrait alors écrire les fonctions d'écriture dans la base et les spécifier à `session.save_handler`

`session.save_path`

définit l'argument qui est passé à la fonction de sauvegarde. Si vous utilisez la sauvegarde par fichier, cet argument est le chemin jusqu'au dossier où les fichiers sont créés. Par défaut, le dossier est /tmp. Si le dossier que vous utilisez a les droits de lecture universels, comme /tmp (valeur par défaut), les autres utilisateurs du serveur peuvent aussi lire ces fichiers, et s'immiscer dans vos sessions.

`session.name`

spécifie le nom de la session, qui sera utilisé comme nom de cookie. Par défaut : PHPSESSID.

`session.auto_start`

indique qu'une session doit commencer automatiquement lors de la premier requête. Par défaut, la valeur est à 0 (inactivé) ; il faut donc utiliser **`session_start()`**.

`session.lifetime`

fixe la durée de vie, en secondes, du cookie envoyé au client. La valeur 0 signifie "jusqu'à ce que le client soit fermé". Par défaut à 0 (inactivé).

`session.serialize_handler`

définit le nom de la fonction qui sera utilisée pour enregistrer et relire les donnés. Actuellement, c'est un format interne de PHP (nom : php) et WDDX (nom : wddx). WDDX n'est utilisable que si PHP a été compilé avec le support WDDX. Par défaut, c'est le mode php qui est sélectionné.

`session.gc_probability`

précise la probabilité que la routine gc (garbage collection) soit lancée, en pourcentage. Par défaut, la valeur est à 1.

`session.gc_maxlifetime`

fixe la durée, en secondes, au-delà de laquelle les données considérées comme inutiles seront supprimées.

`session.referer_check`

représente la sous-chaîne que vous utilisez pour vérifier la provenance de l'internaute. Si l'entête HTTP Referer vous est fournie par le navigateur et que cette sous-chaîne n'est pas trouvée, la session qui vous est fournie sera considérée comme invalide (car provenant probablement d'un autre site que le votre). Par défaut, cette chaîne est vide.

`session.entropy_file`

est le chemin jusqu'à une source externe (fichier) d'entropie, qui sera utilisée lors de la création de l'identifiant de session. Par exemple, /dev/random ou /dev/urandom qui sont disponibles sur de nombreux systèmes UNIX.

`session.entropy_length`

précise le nombre d'octets qui seront lus dans le fichier ci-dessus. Par défaut, 0 (inactivé).

`session.use_cookies`

cookies indique si le module doit utiliser des cookies pour enregistrer l'identifiant de session chez le client. Par défaut, 1 (activé). Ce qui suppose que le client accepte les cookies, ce qui n'est pas acquis ! C'est pourquoi certains serveurs font le choix de ne pas stocker l'identifiant de session sous forme de cookie mais le rajoute systématiquement en paramètre dans toutes les URL.

`session.cookie_path`

spécifie le chemin à utiliser avec session_cookie. Par défaut, /.

`session.cookie_domain`

spécifie le domaine à utiliser avec session_cookie. Par défaut, rien du tout.

`session.cache_limiter`

spécifie le contrôle du cache, à utiliser avec les pages de session (nocache/private/public). Par défaut, nocache.

`session.cache_expire`

spécifie la durée de vie des pages de session cachées, en minutes, mais sans que cela ait d'effets sur le limiteur "nocache". Par défaut, 180.

`session.use_trans_sid`

indique si le support du SID est activé ou pas, lors de la compilation avec l'option --enable-trans-sid. Par défaut, elle vaut 1 (activée).

`url_rewriter.tags`

spécifie si les balises html sont réécrites pour inclure l'identifiant de session si sid est activé. Par défaut, a=href, area=href, frame=src, input=src, form=fakeentry. Permet de passer l'identifiant de session de page en page par l'URI pour parer au refus de cookie, c'est la seule alternative à `session.use_cookies`.

[haut]

Cas particulier de l'hébergeur FREE.FR

Cher free, le répertoire de sessions doit être à la racine de votre compte FTP. Il suffit donc de créer le répertoire **sessions** (au pluriel).

Comment faire cohabiter sur le même serveur deux applications utilisant les sessions ?

Dans le cas où un serveur HTTP héberge plusieurs applications PHP utilisant chacune les sessions, il peut y avoir des problèmes si plusieurs applications utilisent les mêmes variables de sessions (lorsque un utilisateur visite simultanément plusieurs applications).

Pour résoudre ce problème il suffit de définir un nom de session différent pour chacune des applications lors de la création de la session :

```
session_name('appli1');  
session_start();
```

Puis de rappeler ce nom lors de l'utilisation des variables de la session dans l'application.

```
session_name('appli1');  
session_start();
```

Autre solution, au démarrage d'une nouvelle session, pour éviter tout conflit dans l'utilisation des noms de variables de session, spécifier un autre chemin de sauvegarde des données de session avec `session_save_path()`.

[haut]

Ce document est issu de <http://cyberzoide.developpez.com> et reste la propriété exclusive de son auteur. La copie, modification et/ou distribution par quelque moyen que ce soit est soumise à l'obtention préalable de l'autorisation de l'auteur : Hugo ETIEVANT (*cyberzoide at yahoo dot fr*).