

## Php : la gestion des sessions

Formation

**Développeur Web et Web Mobile**

Module : 04

**Développer la partie back-end d'une application web**

Séquence : 03

**Développer une interface utilisateur web dynamique**

Séance : 01


**Développer des scripts serveurs**

Libellé réduit:	PhpSession
Type de document:	Ressource
Version:	1
Date de mise à jour:	13/11/2020

## Sommaire

### Sommaire

I.	Introduction.....	1
II.	Demarrage d'une session.....	1
III.	Sécuriser les identifiants de session.....	2
IV.	La gestion de la durée de vie des sessions.....	3
I.1	l'identifiant de session est transmis par l'URL.....	3
I.2	l'identifiant de session est transmis par cookie.....	3
I.3	role du ramasse-miettes.....	4

	Auteur	Centre de Créteil	Formation	Date Mise à jour	Page 1
	Didier Bonneau	GRN 164	DWWM	15/11/2019	cours php - les sessions.docx

# I. INTRODUCTION

PHP met à disposition un ensemble de fonctions pour manipuler un mécanisme de **session** : un moyen efficace de conserver des données d'un utilisateur entre deux requêtes.

Lors de la première connexion sur un site, il n'y a aucun identifiant transmis dans la requête, le serveur attribuera alors un identifiant unique à l'utilisateur, qui sera retransmis à chaque requête (par le biais d'un cookie ou d'une variable dans l'URL). PHP utilisera cet identifiant pour retrouver les données de l'utilisateur qu'il stocke sur le serveur.

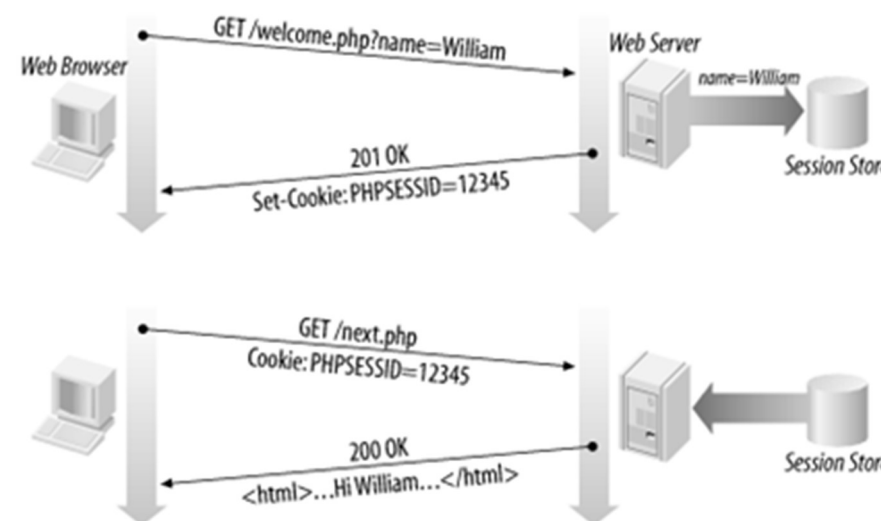
# II. DEMARRAGE D'UNE SESSION

Pour que php gère la session utilisateur lors d'une requête, il faut démarrer la session par la fonction « **session\_start()** » en début de script. Cet appel est implicite si la directive « **session.auto\_start** » du « php.ini » vaut « **on** ».

Si aucun identifiant de session n'a été transmis dans la requête, alors PHP générera une valeur aléatoire renvoyée à l'utilisateur. Si un identifiant existe, PHP remplira la variable globale « **\$\_SESSION** » avec les données enregistrées.

Cette variable est en fait un tableau associatif dont les clés sont les « variables de session ».

Les données de session sont protégées en écriture : cela signifie qu'un seul script à la fois sera en mesure de les modifier. Ceci peut ralentir l'exécution de requêtes simultanées (avec les frames ou Ajax par exemple), l'enregistrement de la session sera retardé en attendant que le ou les autre(s) script(s) soi(en)t terminé(s). Pour limiter cette attente, vous pouvez appeler explicitement la fonction « **session\_write\_close()** » (ou son alias « **session\_commit()** »).



```
<?php
// Initialisation de la session
session_start();
// On teste la variable de session varSession, et
// on l'écrit si elle n'existe pas encore
if (!isset($_SESSION['varSession']))
    $_SESSION['varSession'] = 'valeur';
echo $_SESSION['varSession']; // La variable est
                               // utilisable dans le script courant
?>
```

afpa©	Auteur	Centre de Créteil	Formation	Date Mise à jour	Page 1
	Didier Bonneau	GRN 164	DWWM	15/11/2019	cours php - les sessions.docx

### III. SECURISER LES IDENTIFIANTS DE SESSION

Fondamentalement, l'implémentation des sessions en PHP ne souffre pas de problèmes de sécurité, mais vous devrez néanmoins veiller à la manière dont vous les utiliserez. En effet, dans la mesure où une session est basée sur un identifiant transmis sur Internet entre l'utilisateur et le serveur, cette donnée est très sensible. Si une personne mal intentionnée obtient l'identifiant de session d'un autre utilisateur, il devient assez facile d'usurper son identité.

Il existe plusieurs techniques (simples ou plus compliquées) permettant d'obtenir l'identifiant de session d'un utilisateur :

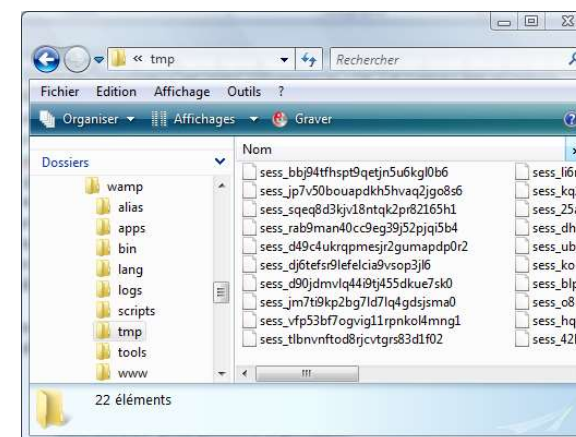
- ✓ la capture : c'est simplement le vol de l'identifiant en le lisant chez l'utilisateur (risque qui doit être contrôlé par l'utilisateur),
- ✓ la divination : l'identifiant est deviné par la personne mal intentionnée (pratiquement impossible),
- ✓ la fixation : le pirate choisit et force l'identifiant de la session (que l'on peut éviter simplement).

D'une manière générale, on évitera de transmettre l'identifiant de session dans l'URL par sécurité (pour prévenir de la fixation). D'ailleurs, c'est assez mauvais pour le référencement du site. Ce comportement peut être désactivé avec la directive « **session.use\_only\_cookies** » à vrai. Bien entendu, un utilisateur interdisant l'utilisation des cookies ne pourra pas utiliser les sessions.

Dans tous les cas, quand le niveau de sensibilité d'une session est modifié (par exemple, quand l'utilisateur s'authentifie ou qu'il obtient de nouveaux droits d'accès), une bonne pratique consiste à générer de nouveau cet identifiant de session avec « **session\_regenerate\_id()** ».

Par défaut, les données de sessions de PHP sont stockées dans un simple fichier de texte (dans le répertoire tmp de wamp ou xampp sous windows).

Tous les scripts PHP peuvent lire ces fichiers car toute exécution de PHP est réalisée par un même utilisateur (à moins que le « **safe\_mode** » soit activé). Sur un hébergeur mutualisé, il est peut-être plus prudent de stocker les données de sessions dans une base de données. Consultez la documentation de la fonction « **session\_set\_save\_handler()** » pour en savoir plus sur la personnalisation de la gestion du stockage des données de sessions.



## IV. LA GESTION DE LA DUREE DE VIE DES SESSIONS

Deux cas : les sessions gérées par URL ou par cookies.

L'un des points les plus délicats à appréhender dans le mécanisme des sessions concerne la durée de vie. Dans cette section, nous allons voir qu'il y a plusieurs facteurs qui permettent de déterminer la durée de vie d'une session.

### I.1 L'IDENTIFIANT DE SESSION EST TRANSMIS PAR L'URL

Dans ce cas, la session existera tant que l'identifiant sera transmis par l'URL.

En fait, même en recopiant la même adresse dans un autre navigateur, la session sera retrouvée.

### I.2 L'IDENTIFIANT DE SESSION EST TRANSMIS PAR COOKIE

Si on utilise un cookie, la durée de vie d'une session dépendra de la durée de vie d'un cookie.

Cette durée est définie à l'aide de la directive « **session.cookie\_lifetime** ». Si cette valeur vaut 0, alors le cookie sera maintenu par le navigateur tant que ce dernier ne sera pas fermé par l'utilisateur.

La plupart des directives de manipulation des sessions définies dans la configuration de PHP peuvent être modifiées dynamiquement pour un script, à condition que ces modifications interviennent avant le démarrage d'une session avec « `session_start()` ». Si le démarrage automatique est actif, vous ne pourrez pas manipuler ces valeurs. Pour pouvoir faire quelques tests, voir la fonction « `session_set_cookie_params()` ».


Le script qui suit affiche la durée de vie restante du cookie de session et une chaîne de caractère définie à la création de la session. Tant que cette valeur ne change pas, cela signifie que la session utilisée est toujours la même.

```
<?php
// Durée de vie de la session (Time To Live)
$ttl = 10;
session_set_cookie_params($ttl);

session_start();

if(!isset($_SESSION['valeur']) ||
!isset($_SESSION['temps'])) {
    $_SESSION['valeur'] = 'valeur' . rand(1, 100);
    $_SESSION['temps'] = time()+$ttl;
}

echo 'TTL de la session ('. $_SESSION['valeur'].') :
'.(($_SESSION['temps'])-time());
?>
```

	Auteur	Centre de Créteil	Formation	Date Mise à jour	Page 3
	Didier Bonneau	GRN 164	DWWM	15/11/2019	cours php - les sessions.docx

### I.3 ROLE DU RAMASSE-MIETTES

Nous avons vu ce qui causait la mort d'une session du côté du client, mais lorsque le cookie est supprimé, ou que l'identifiant n'est plus transmis, les données stockées sur le serveur ne sont pas supprimées pour autant.

PHP a donc mis en place un système de ramasse-miettes (ou garbage collector en anglais) chargé de supprimer ces données persistantes au bout d'un certain temps. Cependant, ce temps n'est pas nécessairement le même que celui de la durée de vie du cookie.

Problème : l'appel du ramasse-miettes s'effectue aléatoirement lors du démarrage de la session.

La durée de vie des données sur le serveur est définie par la directive « session.gc\_maxlifetime ». Si les données ont été supprimées du serveur avant que la session ait expiré chez le client, cette dernière est tout de même réinitialisée. Il faut donc considérer que la durée de vie réelle d'une session est la plus petite des valeurs entre la durée de vie du cookie et la durée de vie des données sur le serveur.

Les données ne seront effectivement supprimées que lorsque le ramasse-miette sera appelé. Cette action est effectuée aléatoirement lors du démarrage de la session, la probabilité de son déclenchement est défini par deux directives : « session.gc\_probability » et « session.gc\_divisor ». On a donc :

probabilité d'appel = gc\_probability/gc\_divisor

Les valeurs par défaut sont respectivement 1 et 100, la probabilité est donc 1/100.

Si vous choisissez de modifier la durée de vie de la session du côté du client (avec « session\_set\_cookie\_params() »), il faudra appliquer les mêmes règles au ramasse-miettes.

Cette opération ne peut être effectuée qu'en modifiant la directive « session.gc\_maxlifetime » de php.ini.

Il est tout à fait possible de faire cette modification avec la fonction « ini\_set() », mais il faudra prendre certaines précautions. En effet : l'appel à « ini\_set() » modifie dynamiquement la configuration, ainsi, lors d'une autre exécution de wamp, PHP considérera la valeur écrite statiquement dans les fichiers de configuration.


La solution la plus simple consiste à choisir un autre répertoire de stockage des données de session, un sous-répertoire par exemple :

```
<?php
$ttl = 3600; // Une heure, en secondes
$new_save_path =
ini_get('session.save_path').'/sousDossier';

session_set_cookie_params($ttl);
ini_set('session.gc_maxlifetime', $ttl);
ini_set('session.save_path', $new_save_path);

session_start();
```

Avec cette solution, le garbage collector appelé dans le contexte d'un autre script supprimera les sessions périmées dans le répertoire défini par session.save\_path, et non le sous répertoire que nous avons choisi. Pensez quand même à créer ce sous-répertoire !

	Auteur	Centre de Créteil	Formation	Date Mise à jour	Page 4
	Didier Bonneau	GRN 164	DWWM	15/11/2019	cours php - les sessions.docx