UNIVERSITÉ DE CAEN **Master IDC**
U.F.R des Sciences **Programmation d'Applications Web**

La gestion de sessions avec PHP

Jean-Marc.Lecarpentier@unicaen.fr

---

1 Les sessions : quel besoin ?

Le protocole HTTP est un protocole "sans état" :
— ne garde aucune trace des requêtes faites au serveur
— exécute chaque requête indépendamment des autres
— impossible de conserver le contexte en cours

Le problème posé vient de la nécessité de conserver des informations sur un utilisateur et/ou ses actions d'une page web à une autre, par exemple :
— identification de l'utilisateur
— statut de l'utilisateur dans le site (visiteur, rédacteur, gestionnaire par ex.)
— parcours de l'utilisateur dans le site
— choix effectués par l'utilisateur (panier d'achats, etc.)

2 Quelles solutions possibles ?

Il existe plusieurs possibilités pour répondre aux besoins exprimés au paragraphe 1 :
— URL longues (méthode GET)
— Champs cachés de formulaires (méthode POST)
— Cookies
— Sessions

La première méthode pose problème car toutes les informations passent dans l'URL et sont donc visibles. Avec les champs cachés de formulaires ou les cookies, les informations sont stockées sur le poste client, d'où le peu de fiabilité de ces méthodes.

Les sessions, elles, stockent les informations sur le disque dur du serveur, et ne sont donc accessibles que par le programme PHP, permettant ainsi de garantir la fiabilité du système (moyennant quelques précautions évidemment, se référer au manuel PHP pour plus de détails).

3 Les sessions

3.1 Principes de fonctionnement

Les sessions offrent la possibilité de stocker des informations sur le disque dur du serveur web, informations qui seront accessibles en lecture/écriture par les programmes PHP.

Lors de l'initialisation d'une session, un identifiant unique (identifiant de session) est généré par le serveur. Cet identifiant est transmis au client (via l'URL ou un cookie), qui le renverra au serveur au changement de page afin de pouvoir accéder aux données le concernant.

Les sessions ont une durée de vie limitée (configurable), c'est à dire qu'en cas d'inactivité pendant un certain temps, les données stockées sur le serveur sont effacées.

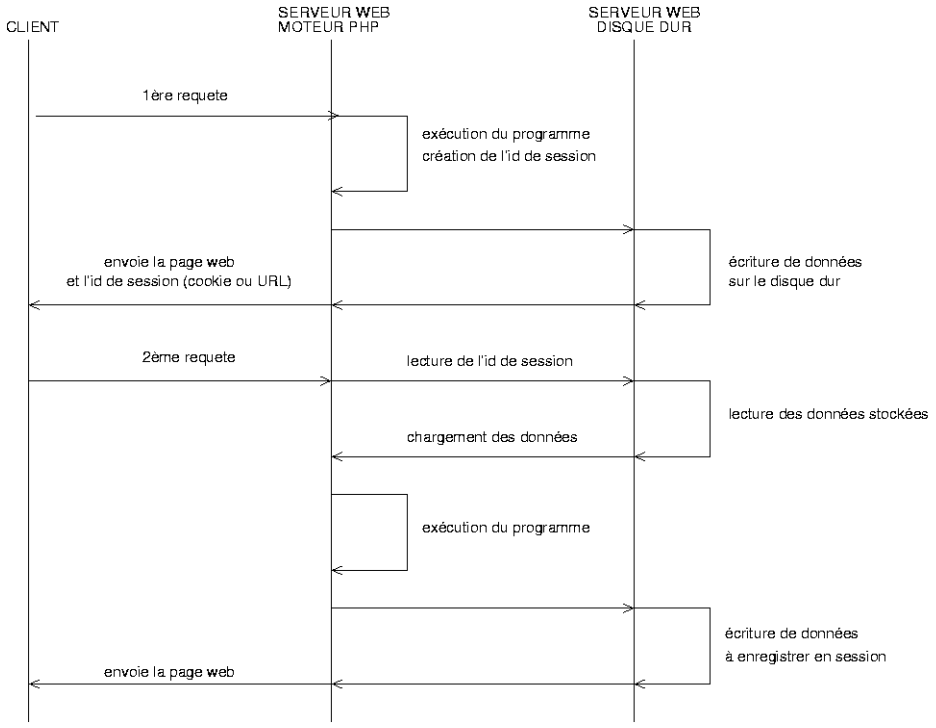

FIGURE 1 – Fonctionnement d'une session

3.2 Cookie ou une URL longue ?

L'identifiant de session est en général stocké dans un cookie qui s'efface lorsque le navigateur est fermé.

Cependant les cookies peuvent être éventuellement refusés par le client, dans ce cas l'identifiant de session ne peut être stocké de cette façon.

Si l'option `session.use_trans_sid` est activée, le serveur détermine automatiquement s'il doit uiliser un cookie ou une URL longue. Par défaut un cookie sera implanté sur le client. Si le client le refuse, alors les URL longues seront alors utilisées, et les URL contenues dans les liens sont réécrites automatiquement pour y ajouter l'identifiant de session. On peut aussi autoriser le fonctionnement des sessions uniquement avec les cookies (option `session.use_only_cookies`).

Remarque 1
La configuration des sessions est visualisable avec l'instruction `phpinfo()`, dans la partie Sessions.

3.3 Le fonctionnement des sessions avec PHP

L'utilisation des fonctionnalités de session doit être explicitement demandé. La fonction `session_start()` réalise cette opération. Si aucune session n'existe, alors une session est créée et un identifiant de session est généré, ainsi qu'un fichier sur le serveur. Si une session existe déjà, alors les variables enregistrées sont chargées en mémoire. Ces variables des session sont stockées dans le tableau associatif "superglobal" `$_SESSION`.

Remarque 2
La configuration de PHP par défaut crée sur le client un cookie nommé `PHPSESSID`. Pour une application réelle il est important de changer ce nom qui ne signifie rien. La fonction `session_name()` permet cela, mais doit être appelée **avant** `session_start()`.

Remarque 3
La lecture ou l'écriture d'un cookie se transmet dans les en-têtes HTTP. Pour cette raison, l'appel à la fonction `session_start()` doit être fait AVANT tout envoi de données au navigateur (même un espace !).

Exemple 1
Le code suivant (avec un seul espace en début de fichier représentés par le souligné)
`_<?php session_start(); ?>` provoquera l'erreur classique suivante :

```
Warning: Cannot send session cache limiter - headers already sent (output
started at /home/jml/public_html/FC2003/sessions/tp/index.php:1) in
/home/jml/public_html/FC2003/sessions/tp/index.php on line 3
```

Une fois la session initialisée, on lit ou écrit les variables de session en utilisant le tableau `$_SESSION`. Seules les variables stockées dans ce tableau seront sauvegardées.

Pour supprimer une variable de session, il suffit d'utiliser la fonction `unset()`.
La fonction `session_destroy()` efface toutes les variables de session.

3.4 Autres fonctions

La fonction `session_encode()` encode sous forme de chaÓne de craractères les variables de la session.

La fonction `session_decode()` fait l'opération inverse.

La fonction `session_set_cookie_params` permet de modifier les paramètres du cookie contenant l'identifiant de session.

3.5 Exemple de script

```
<?php
session_start();

// teste si l'utilisateur est connu ou non
if (!($_SESSION["nom"] && $_SESSION["prenom"])) {
  $message = "Vous n'êtes pas identifié sur le site";
}
else {
  $message = "Bienvenue " . $_SESSION["prenom"] . " " . $_SESSION["nom"];
}

// ... suite du script
?>
```

4 Les objets et les sessions

Les variables stockées en session peuvent être de tout type. Les variables sont sérialisées pour écrire les données dans le fichier de session sous forme de texte. Lors du chargement des données de la session, les variables sont alors désérialisées. Cela ne pose aucun problème pour les types simples (entier, booléen, chaÓne de caractères, etc). En revanche, pour une instance de classe (objet) stockée en session, la désérialisation ne peut s'effectuer correctement que si la déclaration de la classe a été lue auparavant.

En pratique, il faut donc toujours appliquer la règle suivante :

> Inclure les déclarations des classes utilisées par l'application **avant** l'instruction `session_start()`