# PHP 6° PARTIE :
## LES SESSIONS

1. Introduction
2. Identificateur de session
3. Variables de session
4. Client / Serveur
5. Principe
6. Ouverture de session
7. Enregistrement de variables de session
8. Utilisation des variables de session
9. Suppression des variables de session
10. Suppression de session
11. Exemple simple
12. Configuration du contrôle de session
13. Options de configuration
14. Aller plus loin …
15. Les sessions et la sécurité
16. Travaux pratiques

# 1 . Introduction

✓ Les sessions PHP est une notion introduite dans la version 4 de PHP. Ces sessions sont très utiles dans la mesure où elles sont les seules à permettre un suivi fiable du visiteur dans un site.

✓ Les sessions permettent d'enregistrer des variables propres à un utilisateur.

✓ Une session est caractérisée par :

- ❖ Un identificateur de session
- ❖ Des variables  associées à la session

# 2 . Identificateur de session

Une session est créée avec l'appel : **`session_start().`**

*Remarque :* comme pour les *cookies*, la session est initialisée avec l'envoi *header* d'une page.

La fonction `session_start()` essaye de trouver un identificateur de session existant, sinon une nouvelle session est créée et un identificateur unique est assigné par PHP.

Toutes les pages utilisant les sessions doivent donc appeler cette fonction pour indiquer au moteur PHP de récupérer les informations relatives à la session.

La session, à proprement parlé, est stockée côté **serveur** : seul l'identificateur de la session se trouve côté **client**.

La fonction `session_start()` essaye donc de trouver un identificateur de session dans les ***cookies***. Sinon, elle regardera si l'identificateur a été passé en paramètre par la méthode **GET** (directement dans l'URL). Par défaut, l'identificateur de session est passé par la variable **`PHPSESSID`** et à la forme suivante : 2c5lf0fbc54e07c9cl4c7fd82aa2598b

# 3 . Les variables de session

- **Une variable de session est une variable régulière globale qui, quand elle est enregistrée, garde sa valeur sur toutes les pages qui utilisent les sessions PHP4.**

- **Concrètement, une fois qu'une variable de session est créée, il vous suffit de ré-ouvrir la session pour y avoir directement accès, sans même avoir besoin de la déclarer ou de l'importer.**

- **Les variables de sessions sont enregistrées sur le serveur. On n'a donc pas la contrainte des *cookies*.**

- **Cependant, on doit avoir l'identificateur de session pour retrouver les variables de l'utilisateur associées à la session.**

# 4 . Client / Serveur

*En résumé :*

**Côté client**

Un *cookie* ou une variable contenant un numéro vous identifiant : c'est **identificateur** de session.

**Côté serveur**

Un stockage contient les informations sur les **variables de session**.

Par défaut, Le nom du fichier correspond à la valeur de l'identificateur de session assigné.

En fait PHP4 utilise des fichiers pour enregistrer les variables de session mais il est possible d'utiliser une base de données ou une mémoire partagée pour le même résultat.

# 5 . Principe

➢ Les principales étapes d'utilisation d'une session sont les suivantes :

- ❖ ouverture d'une session ;
- ❖ enregistrement des variables de session ;
- ❖ utilisation des variables de session ;
- ❖ suppression des variables de session ;
- ❖ suppression de la session.

# 6 . Ouverture de session

- Il existe trois possibilités pour ouvrir une session.

  - La plus simple, c'est la méthode recommandée, consiste à appeler en début de chaque script la fonction `session_start()` :

```
<?php
session_start();
…
```

Cette fonction détermine s'il existe déjà un ID de session courant ou le crée s'il n'existe pas. Si l'ID est déjà créé, elle charge alors les variables de sessions enregistrées.

- Sinon, vous pouvez utiliser l'ouverture automatique de session :

  - soit en enregistrant directement une variable de session ;
  - soit en configurant PHP avec l'option **`session.auto_start`** dans le fichier **`php.ini`**.

# 7 . Enregistrement de variables de session

Pour qu'une variable puisse être utilisée d'un script à un autre, elle doit être enregistrée en tant que variable de session, par un appel de la fonction `session_register()`.

Par exemple, pour enregistrer la variable `$myvar`, vous pouvez utiliser le code suivant :

```
$myvar = 5;
session_register(”myvar”); //enregistrement de la variable et de sa valeur
```

La variable est ainsi enregistrée jusqu'à la fin de la session, ou jusqu'à son dés-enregistrement manuel.

Vous pouvez enregistrer plusieurs variables à la fois, sous la forme d'une liste de noms de variables séparés par des virgules, comme suit :

```
session_register(”myvar1”, ”myvar2”);
```

# 8 . Utilisation des variables de session

- Suivant la version de PHP et les options configurées dans php.ini :

  - Si l'option **`register_globals`** a été activée, vous pouvez accéder à la variable *via* son nom (par exemple **`$myvar`**).
  - Si l'option **`track_vars`** est activée, vous pouvez accéder à la variable par le biais du tableau associatif **`$HTTP_SESSION_VARS`** (par exemple `$HTTP_SESSION_VARS["myvar"]`).
  - A terme ces options vont disparaître pour des raisons de sécurité. La nouvelle implémentation est la suivante : le tableau **`$_SESSION`** (exemple `$_SESSION["myvar"]`).

Remarques :

- Une variable de session ne peut être "écrasée" par des données GET ou POST. Cette limitation est en réalité une bonne mesure de sécurité, mais qu'il ne faut pas oublier lorsque vous programmez.

- Pour déterminer si une variable est une variable de session enregistrée, il faut utiliser la fonction `session_is_registered()`, comme suit :

```
$result = session_is_registered("myvar") ;
```

# 9 . Suppression des variables de session

Une fois que vous en avez terminé avec une variable de session, vous pouvez la supprimer en invoquant la fonction `session_unregister()`, comme suit :

```
session_unregister(”myvar”) ;
```

Là encore, cette fonction prend comme argument le nom de la variable à dès-enregistrer, sous la forme d’une chaîne sans préfixe `$`.

La fonction `session_unregister()` ne peut dès-enregistrer qu’une seule variable de session à la fois (contrairement à la fonction `session_register()`).

Vous pouvez toutefois obtenir le désenregistrement de toutes les variables de session courantes au moyen de la fonction `session_unset()`.

# 10 . Suppression de session

Une fois que vous en avez terminé avec une session, et après avoir dès-enregistré toutes les variables, il faut invoquer la fonction :

```
session_destroy(); //pour supprimer l'ID de la session.
```

*En résumé :*

- `session_start()` : Initialise une session
- `session_destroy()` : Détruit les données de la session courante
- `session_register(string)` : Enregistre la variable `string`
- `session_unregister(string)` : Efface la variable `string`
- `session_is_registered(string)` : Renvoie un booléen indiquant si la variable existe

# 11 . Exemple simple de session

Nous allons faire un marché, avec des pommes des poires et des bananes à acheter, on pourra le acheter une à une et supprimer toutes les pommes, toutes les poires ou toutes les bananes, on pourra également vider complètement son panier.

```
<?php
session_start();
session_register("pommes");
session_register("poires");
session_register("bananes");

if ($action=="destroy") {
        session_destroy(); header("location: sample_session.php"); }

switch ($raz)
{
        case "pomme":     session_unregister("pommes");
                          header("location: sample_session.php");
                          break;
        case "poire":     session_unregister("poires");
                          header("location: sample_session.php");
                          break;
        case "banane":    session_unregister("bananes");
                          header("location: sample_session.php");
                          break;
}
```

# 11 . Exemple simple de session

```
switch ($achat)
{          case "pomme":        $pommes++; break;
           case "poire":        $poires++; break;
           case "banane":       $bananes++;break;
}

echo "Nb Pommes:$pommes <a href=sample_session.php?achat=pomme>Acheter une pomme</a>
<a href= sample_session.php?raz=pomme>RAZ</a><br>";

echo "Nb Poires:$poires <a href= sample_session.php?achat=poire>Acheter une poire</a>
<a href= sample_session.php?raz=poire>RAZ</a><br>";

echo "Nb Bananes:$bananes <a href= sample_session.php?achat=banane>Acheter une banane</a>
<a href= sample_session.php?raz=banane>RAZ</a><br>";

echo "<br><a href= sample_session.php?action=destroy>Détruire la session</a></br>";
?>
```

Remarques:

Au début on initialise les variables, il ne doit pas y avoir d'envoi de données avant <?php.
Ensuite on étudie la cas où on a cliqué sur "Détruire la session", si c'est le cas on détruit la session et on recharge la page pour prendre en compte ce changement.
Dans le cas d'une remise à 0 d'un des compteur, on désenregistre la variable et on recharge la page.
Dans le cas d'un achat, on incrémente l'un des fruits, ce qui sera répercuté sur la donnée de session.

# 12 . Configuration du contrôle de session

PHP est capable de faire transiter l'identificateur de session de pages en pages de manière transparente si PHP a été compilé avec l'option `enable-trans-sid` (il faut aussi `session.use_trans_sid = 1` dans le fichier `php.ini` ).

Le nom de la session est indiqué dans `php.ini` par `"session.name="` (généralement `PHPSESSID` pour connaître la valeur il suffit d'exécuter `<? php phpinfo() ?>` ) l'instruction `<php echo $PHPSESSID; ?>` affiche l'identificateur de session.

Si l'identificateur de session n'est pas transité par défaut, il suffit d'appeler toute les pages en ajoutent l'argument PHPSESSID=$PHPSESSID.

# 13 . Options de configuration

Vous pouvez optimiser et sécuriser vos sessions en configurant le 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.

**`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`.

**`session.name`** spécifie le nom de la session, qui sera utilisé comme nom de *cookie.* Par défaut, ce sera PHPSESSID

**`session.auto_start`** indique qu'une session doit commencer automatiquement lors de fa première requête. Par défaut, la valeur est à 0 (inactivé).

**`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 données.

**`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`** détermine si l'identifiant de session (session id) utilisé par des sites externes seront éliminés. Si les identifiants de session sont propagés avec la méthode des URL, des utilisateurs qui n'en connaîtraient pas l'utilité risquent de divulguer ces valeurs, et cela mènera à des problèmes de sécurité. Cette option y remédie. Par défaut : 0.

# 13 . Options de configuration

**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** indique si le module doit utiliser des *cookies* pour enregistrer l'identifiant de session chez le client. Par défaut 1 (activé).

**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.cookie_lifetime** fixe la durée maximale du cookie de session sur la machine utilisateur. La valeur 0, qui est la valeur par défaut, spécifie que le cookie doit expirer à la fermeture du navigateur WEB.

**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).

# 14 . Aller plus loin ...

- **Session et base de donnée**

Pour les plus expérimentés, il est possible de redéfinir les fonctions qui servent à enregistrer les données ainsi, au lieu de stocker les variables de sessions dans un fichier on peut stocker ces variables directement dans une base de données. La fonction s'appelle : `session_set_save_handler`.

- **PHPLib**

Dans les version précédentes de PHP, le contrôle de session était supporté par **PHPLib**, la bibliothèque de base de PHP, qui demeure un outil très utile avec PHP4.

Pour obtenir des informations sur cette bibliothèque : http://phplib.netuse.de/index.php3

Depuis la version 4 de PHP, PHP comprend des fonctions natives de contrôle de session. Ces fonctions sont identiques aux fonctions correspondantes de PHPLib, toutefois PHPLib offre des fonctionnalités plus étendues. Donc, si les fonctions natives de PHP ne correspondent plus à vos besoins, il vous faudra examiner les possibilités de PHPLib.

# 15 . Les sessions et la sécurité

- **Répertoire de sauvegarde**

Si le dossier que vous utilisez a les droits de lecture universels, comme `/tmp` (qui est la valeur par défaut), les autres utilisateurs du serveur peuvent aussi lire ces fichiers, et s'immiscer dans vos sessions.

Donc, il est important, pour des applications sécurisées de ne pas définir le répertoire de sauvegarde des sessions comme `/tmp`.

- **Durée de validité de la session**

La clé de session pouvant être présente dans l'URL, un risque existe si un utilisateur copie et envoie cette URL à une seconde personne. Si cette personne l'utilise alors que la session n'est pas encore terminée, les variables du premier utilisateur peuvent être alors utilisées par le second.

# 16 . Travaux pratiques

1. Tester l'exemple du cours (poire-pommes-bananes).

2. Ecrire l'application qui réalisera les fonctions de base suivantes : le suivi des utilisateurs après leur authentification via un mécanisme de login et de session. On conservera dans la session le nom d'utilisateur.

Cette application se composera de trois scripts :

**`main.php`** : fournit un formulaire d'ouverture de session avec authentification (login + mot de passe) pour les membres. Ce script contient aussi la page d'accueil qui se résumera à l'affichage d'un message de bienvenue personnalisé. En cas d'échec, on ré-affichera le formulaire d'identification en précisant le message d'erreur. Si la session n'est pas ouverte et on l'indiquera.

**`members.php`** : affiche des informations à l'intention des utilisateurs ayant réussi à ouvrir une session avec authentification (login + mot de passe corrects). Sinon, on affichera un message d'espace réservé aux membres.

**`logout.php`** : ferme la session ouverte par un membre.

**Remarques :**

Utiliser une table utilisateurs pour le stockage des comptes (userid, login, password).

La fonction crypt() permet le cryptage des mots de passe.