### **Une session c'est quoi ?**

Une session est un mécanisme technique permettant de sauvegarder **temporairement sur le serveur** des informations relatives à un internaute. Ce système a notamment été inventé pour palier au fait que le protocole HTTP agit en mode **non connecté**. A chaque ouverture de nouvelle session, l'internaute se voit attribué un identifiant de session. Cet identifiant peut être transmis soit en GET (PHPSESSID ajouté à la fin de l'url), POST ou Cookie (cookie sur poste client) selon la configuration du serveur. Les informations seront quant à elles transférées de page en page par le serveur et non par le client. Ainsi, la sécurité et l'intégrité des données s'en voient améliorées ainsi que leur disponibilité tout au long de la session. Une session peut contenir tout type de données : nombre, chaîne de caractères et même un tableau.

Contrairement à une base de données ou un système de fichiers, la session conserve les informations pendant quelques minutes. Cette durée dépend de la configuration du serveur mais est généralement fixée à 24 minutes par défaut. Le serveur crée des fichiers stockés dans un répertoire particulier.

Les sessions sont particulièrement utilisées pour ce type d'applications :

* Les espaces membres et accès sécurisés avec authentification.
* Gestion d'un caddie sur un site de vente en ligne.
* Formulaires éclatés sur plusieurs pages.
* Stockage d'informations relatives à la navigation de l'utilisateur (thème préféré, langues...).

La théorie, c'est bien beau mais en pratique comment ça se passe ? La partie suivante explique l'initialisation et la restauration d'une session ouverte.

### **Initialisation (et restauration) d'une session**

PHP introduit nativement une unique fonction permettant de démarrer ou de continuer une session. Il s'agit de [session\_start()](http://fr.php.net/session-start). Cette fonction ne prend pas de paramètre et renvoit toujours **true**. Elle vérifie l'état de la session courante. Si elle est inexistante, alors le serveur la crée sinon il la poursuit.

**Initialisation et restauration d'une session**

**<?php**

[session\_start](http://www.php.net/session_start)();

**?>**

Dans le cas d'une utilisation des sessions avec les cookies, la fonction session\_start() doit obligatoirement être appellée avant tout envoi au navigateur sous peine de voir afficher les fameuses erreurs :  
  
*"Cannot modify header information - headers already sent by ..."* ou *"Cannot send session cookie - headers already sent by ..."*. Cela est du au fait que PHP ne peut plus envoyer de cookie à l'utilisateur car il y'a déjà eu une sortie au navigateur (echo(), print(), espace blanc, tag html...).

***Note :****il faut appeller session\_start() sur chaque page utilisant le système de session.*

### **Lecture et écriture d'une session**

#### *Le tableau $\_SESSION*

Lorsqu'une session est créée, elle est par défaut vide. Elle n'a donc aucun intérêt. Il faut donc lui attribuer des valeurs à sauvegarder temporairement. Pour cela, le langage PHP met en place le tableau superglobal **$\_SESSION**. Le terme *superglobal* signifie que le tableau a une visibilité maximale dans les scripts. C'est à dire que l'on peut y faire référence de manière globale comme locale dans une fonction utilisateur sans avoir à le passer en paramètre. Le tableau **$\_SESSION** peut être indexé numériquement mais aussi associativement. En règle générale, on préfère la seconde afin de pouvoir donner des noms de variables de session clairs et porteurs de sens.

#### *L'écriture de session*

Pour enregistrer une nouvelle variable de session, c'est tout simple. Il suffit juste d'ajouter un couple clé / valeur au tableau $\_SESSION comme l'illustre l'exemple suivant.

**Déclaration et initialisation d'une variable une session**

**<?php**

*// Démarrage ou restauration de la session*

[session\_start](http://www.php.net/session_start)();

*// Ecriture d'une nouvelle valeur dans le tableau de session*

$\_SESSION['login'] = 'Dupond';

**?>**

Le tableau $\_SESSION, qui était vide jusqu'à présent, s'est agrandi dynamiquement et contient maintenant une valeur (**Dupond**) à la clé associative **login**. Une variable de session est alors créée.

***:****à place de la chaîne de caractères « Dupond », nous aurions pu mettre un nombre, un booléen ou encore un tableau par exemple.*

#### *Lecture d'une variable de session*

Après l'écriture, c'est au tour de la lecture. Il n'y a rien de plus simple. Pour lire la valeur d'une variable de session, il faut tout simplement appeler le tableau de session avec la clé concernée. L'exemple ci-dessous illustre tout ça.

**Lecture d'une variable de session**

**<?php**

*// Démarrage ou restauration de la session*

[session\_start](http://www.php.net/session_start)();

*// Lecture d'une valeur du tableau de session*

[echo](http://www.php.net/echo) $\_SESSION['login'];

**?>**

Cette instruction aura pour effet d'afficher à l'écran la chaîne de caractères **Dupond**.

### **Destruction d'une session**

Comme cela a été évoqué plus haut, le serveur détruit lui-même la session au bout d'un certain temps si la session n'a pas été renouvelée. En revanche, il est possible de forcer sa destruction au moyen de la fonction [session\_destroy()](http://fr.php.net/session-destroy). Cela permet par exemple aux webmasters de proposer une page de déconnexion aux membres loggués à leur espace personnel. Cependant, l'utilisation de session\_destroy() seule n'est pas très "propre". Le code suivant présente une manière plus correcte de mettre fin à une session.

**Destruction propre et complète d'une session**

**<?php**

*// Démarrage ou restauration de la session*

[session\_start](http://www.php.net/session_start)();

*// Réinitialisation du tableau de session*

*// On le vide intégralement*

$\_SESSION = [array](http://www.php.net/array)();

*// Destruction de la session*

[session\_destroy](http://www.php.net/session_destroy)();

*// Destruction du tableau de session*

[unset](http://www.php.net/unset)($\_SESSION);

**?>**

Pour être convaincu de la destruction de la session, il suffit juste d'essayer d'afficher le contenu du tableau de session au moyen de la fonction [print\_r()](http://fr.php.net/print-r).

### **Configuration des sessions sur le serveur**

Une session ne reste ouverte que pendant un certain temps. Tout au plus ce sera celle indiquée par la directive *session.gc\_maxlifetime* du php.ini, entre deux clics consécutifs du client. Il est recommandé de ne pas augmenter la valeur inscrite par défaut. Mais pourquoi ? Tout simplement parce que si la session à une durée de vie plus importante, on s'expose à des risques de piratage par *vol de session* notamment (voir les liens annexes en fin de tutoriel pour plus d'informations).

Pour les mêmes raisons de sécurité, il est conseillé de configurer le serveur de la façon suivante :

**Configuration de PHP recommandée pour les sessions**

session.use\_cookies 1

session.use\_only\_cookies 1

session.use\_trans\_sid 0

Cette configuration implique néanmoins une restriction totale pour les personnes n'acceptant pas les cookies. Ci-dessous, la signification dans le même ordre des 3 lignes de configuration précédentes.

* L'identifiant de session est transmis par un cookie.
* Seul le cookie peut transmettre l'identifiant de session.
* Le PHPSESSID transmis dans l'url est strictement refusé.

***Variables de session pour gérer une page protégée***

***Présentation du cas pratique***

Ici un cas concret d'utilisation des sessions. Il s'agit d'un accès restreint basique. Seul un utilisateur est autorisé à être loggué mais cet exemple est à la base de la création d'un espace membre. C'est exactement la même chose. Pour réaliser tout ça, nous aurons besoin de 2 fichiers : le formulaire accompagné de son script de login, et la page protégée. Commençons par le formulaire de login. Le code étant commenté, il n'y aura pas plus d'explications.

***Formulaire d'authentification : authentification.php***

**Cas pratique : formulaire d'identification à un espace membre**

**<?php**

*// Definition des constantes et variables*

[define](http://www.php.net/define)('LOGIN','toto');

[define](http://www.php.net/define)('PASSWORD','tata');

$errorMessage = '';

*// Test de l'envoi du formulaire*

if(![empty](http://www.php.net/empty)($\_POST))

{

*// Les identifiants sont transmis ?*

if(![empty](http://www.php.net/empty)($\_POST['login']) && ![empty](http://www.php.net/empty)($\_POST['password']))

{

*// Sont-ils les mêmes que les constantes ?*

if($\_POST['login'] !== LOGIN)

{

$errorMessage = 'Mauvais login !';

}

elseif($\_POST['password'] !== PASSWORD)

{

$errorMessage = 'Mauvais password !';

}

else

{

*// On ouvre la session*

[session\_start](http://www.php.net/session_start)();

*// On enregistre le login en session*

$\_SESSION['login'] = LOGIN;

*// On redirige vers le fichier admin.php*

[header](http://www.php.net/header)('Location: http://www.monsite.com/admin.php');

[exit](http://www.php.net/exit)();

}

}

else

{

$errorMessage = 'Veuillez inscrire vos identifiants svp !';

}

}

**?>**

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

[**<html**](http://december.com/html/4/element/html.html) xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr"**>**

[**<head>**](http://december.com/html/4/element/head.html)

[**<title>**](http://december.com/html/4/element/title.html)Formulaire d'authentification**</title>**

**</head>**

[**<body>**](http://december.com/html/4/element/body.html)

[**<form**](http://december.com/html/4/element/form.html) action="<?php echo htmlspecialchars($\_SERVER['PHP\_SELF']); ?>" method="post">

<fieldset>

<legend>Identifiez-vous</legend>

**<?php**

*// Rencontre-t-on une erreur ?*

if(![empty](http://www.php.net/empty)($errorMessage))

{

[echo](http://www.php.net/echo) '<p>', [htmlspecialchars](http://www.php.net/htmlspecialchars)($errorMessage) ,'</p>';

}

**?>**

**<p>**

[**<label**](http://december.com/html/4/element/label.html) for="login"**>**Login :**</label>**

[**<input**](http://december.com/html/4/element/input.html) type="text" name="login" id="login" value="" /**>**

**</p>**

[**<p>**](http://december.com/html/4/element/p.html)

[**<label**](http://december.com/html/4/element/label.html) for="password"**>**Password :**</label>**

[**<input**](http://december.com/html/4/element/input.html) type="password" name="password" id="password" value="" /**>**

[**<input**](http://december.com/html/4/element/input.html) type="submit" name="submit" value="Se logguer" /**>**

**</p>**

**</fieldset>**

**</form>**

**</body>**

**</html>**

***Exemple de page protégée : admin.php***

**Exemple de page sécurisée avec les sessions**

**<?php**

*// On prolonge la session*

[session\_start](http://www.php.net/session_start)();

*// On teste si la variable de session existe et contient une valeur*

if([empty](http://www.php.net/empty)($\_SESSION['login']))

{

*// Si inexistante ou nulle, on redirige vers le formulaire de login*

[header](http://www.php.net/header)('Location: http://www.monsite.com/authentification.php');

[exit](http://www.php.net/exit)();

}

**?>**

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

[**<html**](http://december.com/html/4/element/html.html) xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr"**>**

[**<head>**](http://december.com/html/4/element/head.html)

[**<title>**](http://december.com/html/4/element/title.html)Administration**</title>**

**</head>**

[**<body>**](http://december.com/html/4/element/body.html)

**<?php**

*// Ici on est bien loggué, on affiche un message*

[echo](http://www.php.net/echo) 'Bienvenue ', $\_SESSION['login'];

**?>**

**</body>**

**</html>**

**Petite explication** : pour protéger chacune des pages de l'administration, il faut absolument ajouter en tête de fichier le premier script entièrement. Celui-ci redirige instantanément l'utilisateur s'il n'est pas convenablement loggué. Sinon il affiche la page Web.