

Utiliser une base de données pour sécuriser vos sessions

par **Adrien Pellegrini** ([Autre articles](#))

Date de publication : 28/08/2006

Dernière mise à jour :

Les sessions étant très utilisées de nos jours, il ne faut jamais négliger le côté sécurité. Afin de mieux sécuriser les sessions, vous pouvez stocker celles-ci dans la base de données.

- 1 - Introduction
- 2 - Gestion du script
 - 2.1 - L'exemple
 - 2.2 - Comment ça marche ?
 - 2.3 - Les différentes fonctions
- 3 - La base de données
 - 3.1 - Table 'site_session'
 - 3.2 - Table 'site_user'
- 4 - Les Fonctions
 - 4.1 - Fonction openSession()
 - 4.2 - Fonction getLoginCheck()
 - 4.3 - Fonction getUserInfo()
 - 4.4 - Fonction getFormView()
 - 4.5 - Fonction getMsg()
 - 4.6 - Fonction dbClean()
 - 4.7 - Fonction closeSession()
- 5 - Mise en application des fonctions
 - 5.1 - Le Formulaire d'identification
 - 5.2 - Les autres pages du site
- 6 - Conclusion
 - 6.1 - Note générale
 - 6.2 - Rappel
 - 6.3 - Notes
 - 6.4 - Remerciements

1 - Introduction

Les sessions créées sur un site sont enregistrées dans un dossier sur le serveur nommé par défaut /tmp. Si par mégarde le serveur venait à être mal configuré, tout le monde hébergé sur ce serveur aurait accès à ce répertoire et donc aux informations contenues dans les sessions.

Si vous utilisez une base de données pour les stocker, ce problème disparaît.

2 - Gestion du script

2.1 - L'exemple

Je vais prendre comme exemple une page d'identification réduite au plus simple. Elle se limitera à un champ 'Username', un 'Password' et un bouton 'submit'. Rien de bien extraordinaire !

2.2 - Comment ça marche ?

La variable de session ne va servir qu'à propager l'identifiant de session d'une page à l'autre et rien d'autre. Aucune autre information ne sera enregistrée dans la variable de session.

La base de données va permettre de stocker quelques informations supplémentaires concernant l'utilisateur et bien sûr l'identifiant de session. Je détaillerai tout cela par la suite.

2.3 - Les différentes fonctions

| Fonctions | Description |
|-----------------|--|
| init_session() | Initialise la session et met à jour la base de données |
| getLoginCheck() | Teste si le couple username/password est correct |
| getUserInfo() | Récupère les informations concernant l'utilisateur |
| getFormView() | Affiche le formulaire d'identification |
| getMsg() | Affiche un message en cas de succès d'identification ou déjà 'loggé' |
| dbClean() | Supprime automatiquement les enregistrements vieux |
| closeSession() | Permet de se deconnecter |

3 - La base de données

3.1 - Table 'site_session'

Explication des champs :

- champ **sid** : contient l'id de session pour la session en cours
- champ **userid** : contient l'id de l'utilisateur (cet id est lié à la table 'site_user')
- champ **last_modified** : est mis à jour automatiquement dès qu'il y a une modification (création) de la ligne dans la table
- champ **ip** : contient l'ip de l'utilisateur (peut servir pour des mesures de sécurité pour tester si l'utilisateur est toujours le même; pas toujours très fiable)
- champ **browser** : contient le nom du navigateur de l'utilisateur (peut aussi servir pour des mesures de sécurité; un utilisateur ne changera pas de navigateur entre 2 pages du site)

Le champ *ip* et *browser* sont cités à titre indicatifs.

| Champ | Type | Attribut | Défaut |
|---------------|--------------|-----------------------------|-------------------|
| sid | text | | |
| userid | mediumint(9) | | |
| last_modified | timestamp | ON UPDATE CURRENT_TIMESTAMP | CURRENT_TIMESTAMP |
| ip | varchar(15) | | |
| browser | varchar(50) | | |

Code SQL de la table

```
CREATE TABLE `site_session` (
  `sid` text NOT NULL,
  `userid` mediumint(9) NOT NULL,
  `last_modified` timestamp NOT NULL default CURRENT_TIMESTAMP on update CURRENT_TIMESTAMP,
  `ip` varchar(15) NOT NULL,
  `browser` varchar(50) NOT NULL
)
```

3.2 - Table 'site_user'

| Champ | Type | Attribut | Extra |
|----------|--------------|----------|----------------|
| id | mediumint(9) | UNSIGNED | auto_increment |
| username | varchar(50) | | |
| password | varchar(32) | | |
| ... | ... | | |

Code SQL de la table

```
CREATE TABLE `site_user` (
  `id` mediumint(9) unsigned NOT NULL auto_increment,
  `active` text NOT NULL,
  `date` datetime NOT NULL,
  `last_modified` timestamp NOT NULL default CURRENT_TIMESTAMP on update CURRENT_TIMESTAMP,
  `username` varchar(50) NOT NULL,
  `password` varchar(32) NOT NULL,
  `email` varchar(100) NOT NULL,
  `country` varchar(50) NOT NULL,
  `gender` varchar(1) NOT NULL,
  `birthday` date NOT NULL,
  `avatar` varchar(255) NOT NULL,
  PRIMARY KEY (`id`)
```


Code SQL de la table

```
)
```


4 - Les Fonctions

4.1 - Fonction openSession()

Cette fonction va supprimer l'ancien enregistrement de la session qui correspond au même *userid*. Ensuite, elle va re-générer l'identifiant de session (pour des mesures de sécurité). Et va insérer le nouvel identifiant dans la base de données.

Vous pouvez trouver dans ce code une utilisation des fonctions `getIP()` et `getBrowser()`. Ces deux fonctions ne sont pas décrites ici mais vous pourrez en trouver un équivalent sur [developpez.com](#) (dans les Sources de PHP et dans la FAQ PHP).

fonction openSession()

```
function openSession($userid) {
    global $db;

    // On supprime la session en cours
    $deleteSQL = $db->prepare('DELETE FROM '.PREFIX_DB_SITE.'.session
        WHERE userid = :userid');
    $deleteSQL->execute(array(':userid' => $userid));

    // Re-génération du sid
    session_regenerate_id();

    // On insère le nouvel id de session dans la db
    $insertSQL = $db->prepare('INSERT INTO '.PREFIX_DB_SITE.'.session (sid, userid, ip, browser)
        VALUES (:sid, :userid, :ip, :browser)');
    $insertSQL->execute(array(':sid' => session_id(),
        ':userid' => $userid,
        ':ip' => getIP(),
        ':browser' => getBrowser(),
    ));

    return TRUE;
}
```

4.2 - Fonction getLoginCheck()

Cette fonction va vérifier si le couple username/password est correct.

Elle n'a pas d'intérêt direct avec ce tutorial.

fonction getLoginCheck()

```
function getLoginCheck($username, $password) {
    global $db;

    $username = addslashes(trim($username));
    $password = md5(addslashes(trim($password)));

    // Petite vérification pour éviter une requête sql
    if (empty($username) && empty($password)) {
        return FALSE;
    } else {
        $sql = $db->prepare('SELECT id FROM '.PREFIX_DB_SITE.'.user
            WHERE username = :username
            AND password = :password ');
    }
}
```



```
function getLoginCheck()
{
    $sql->execute(array(':username' => $username,
        ':password' => $password,
    ));

    $userdata = $sql->fetch(PDO::FETCH_ASSOC);

    if (!empty($userdata)) {
        $userdata = $userdata['id'];
    } else {
        $userdata = FALSE;
    }

    return $userdata;
}
```

4.3 - Fonction getUserInfo()

Récupère les informations contenues dans la table 'site_users' en fonction de l'id de session.

```
function getUserInfo()
{
    global $db;

    $sql = $db->prepare('SELECT u.id, u.active, u.date, u.last_modified, u.username, u.country,
        u.gender, u.birthday, u.avatar
    FROM site_session s
    INNER JOIN site_user u
    ON s.sid = :sid
    AND s.userid = u.id ');

    $sql->execute( array(':sid' => session_id()) );

    $userdata = $sql->fetch(PDO::FETCH_ASSOC);

    if (empty($userdata)) {
        $userdata = FALSE;
    }

    return $userdata;
}
```

4.4 - Fonction getFormView()

Cette fonction affiche le formulaire d'identification.

Comme la fonction *getLoginCheck()*, il n'y a pas de rapport direct avec ce tutorial.

```
function getFormView()
{
    $view = '<form enctype="multipart/form-data" action="'. $_SERVER['PHP_SELF'] .'" method="post"
    id="form">'. "\n";
    $view .= '<label for="username">Username      </label>'.textfield('username'). "\n";
    $view .= '<label for="password">Password      </label>'.textfield('password'). "\n";
    $view .= button('subit', 'Envoyer'). "\n";
    $view .= '</form>';

    echo $view;
}
```



```
function getFormView()
{
}
```

4.5 - Fonction getMsg()

Cette fonction affiche tout simplement un message de bienvenue en cas de succès d'identification.

Comme les fonctions `getLoginCheck()` et `getFormView`, il n'y a pas de rapport direct avec ce tutorial.

```
function getMsg()

function getMsg($username) {
    $view = $username.' ' ;
    if (empty($_POST['closeSession'])) {
        $view .= '<form enctype="multipart/form-data" action="'. $_SERVER['PHP_SELF']. ' " method="post"
id="form">'. "\n";
        $view .= button('closeSession', 'Fermer la session'). "\n";
        $view .= '</form>';
    } else {
        closeSession();
    }

    return $view;
}
```

4.6 - Fonction dbClean()

Cette fonction permet de supprimer à chaque succès d'identification les entrées vieilles d'au moins 2 jours.

```
function dbClean()

function dbClean() {
    global $db;

    $limit = date('Y-m-d H-i-s', mktime(0, 0, 0, date('m'), date('d') - 2, date('Y')));

    $cleanSQL = $db->prepare('DELETE FROM '.PREFIX_DB_SITE.'.session
        WHERE last_modified < :limit');
    $cleanSQL->execute( array(':limit' => $limit) );
}
```

4.7 - Fonction closeSession()

Cette fonction permet de fermer une session en cours en supprimant son entrée dans la base de données et actualise la page.

```
function closeSession()

function closeSession() {
    global $db;

    $deleteSQL = $db->prepare('DELETE FROM '.PREFIX_DB_SITE.'.session
        WHERE sid = :sid');
    $deleteSQL->execute( array(':sid' => session_id()) );

    echo '<meta http-equiv="Refresh" content="0;'. $_SERVER['PHP_SELF']. '>';
}
```


5 - Mise en application des fonctions

5.1 - Le Formulaire d'identification

Nous avons désormais tout le nécessaire pour écrire le script général qui va vérifier d'abord si une session existe et ensuite si elle est enregistrée dans la base de données. Si aucune session n'est présente dans la base de données, le formulaire sera affiché.

Quand le couple username/password est correct, on enregistre la session dans la base de données (après avoir supprimé la session existante).

Script général

```
if ($userdata = getUserinfo()) {
    openSession($userdata['id']);
    getMsg($userdata['username']);
} else {
    if (!$_POST) {
        // Affiche le formulaire
        getFormView();
    } else {
        // Vérifie si le couple username/password est correct
        if (!$loginCheck = getLoginCheck($_POST['username'], $_POST['password'])) {
            // On réaffiche le form
            getFormView();
            echo 'pas bon';
        } else {
            // On nettoie la db
            dbClean();

            // On initialise la session avec le userid récupérer
            // par loginCheck
            openSession($loginCheck);
            $userdata = getUserinfo();

            if (empty($userdata)) {
                die('erreur');
            }

            getMsg($userdata['username']);
        }
    }
}
```

5.2 - Les autres pages du site

Le but des sessions étant de garder les informations d'une page à l'autre, pour récupérer ces informations, on devra désormais faire appel à la fonction *getInfoUser()*.

Par exemple, si vous avez une architecture du site qui ressemble à:

- 1 common.php
- 2 index.php
- 3 test.php

Avec *common.php* qui va inclure tous les script nécessaires en haut de **chaque page**.

index.php qui contiendra le formulaire d'identification.

Et *test.php* que je vais vous expliquer ci-dessous.

Pour récupérer les informations contenues dans la session vous devez, comme je l'ai mentionné plus haut, utiliser la fonction *getUserInfo()* de cette façon-ci dans *common.php*:

common.php

```
if (!$userdata = getUserInfo()) {  
    $userdata = array();  
}
```

Grâce à cette ligne, vous pourrez avoir toutes les informations que vous voudrez simplement en incluant le fichier *common.php*.

Par exemple, le fichier *test.php* :

test.php

```
require_once 'common.php';  
  
if (!empty($userdata['username'])) {  
    echo 'Bonjour ' . $userdata['username'];  
} else {  
    echo 'Bonjour Anonyme';  
}
```


6 - Conclusion





6.1 - Note générale

Mettre des sessions dans une base de données est plus lourd à mettre en oeuvre, mais c'est aussi (et surtout) une méthode bien plus sécurisée.

Avec ce système, la base de données est constamment utilisée : il est bon de la vider de temps à autre afin de se débarrasser des sessions inactives (ce que le serveur Web fait automatiquement pour le stockage classique des sessions).

6.2 - Rappel

Si vous n'êtes pas familier avec les sessions, je vous conseille de lire ces quelques tutoriaux :

- 1  <http://julp.developpez.com/php/les-sessions/>
- 2  <http://beaussier.developpez.com/articles/php/session/>
- 3  <http://cyberzoide.developpez.com/php4/faqsession/>
- 4  [Doc PHP](#)

6.3 - Notes

Pour la connexion à la base de données et les requêtes SQL, j'ai utilisé **l'extension PHP Data Objects (PDO)**. Cette extension nécessite PHP5.

6.4 - Remerciements

Tous mes remerciements à **Yogui** pour sa relecture.

