# DBProtect V1.2

*Fonctionnement détaillé*

**Caractéristiques :**

- Données en BDD MySQL
- Mots de passe chiffrés par md5
- Protection contre l'injection SQL
- Sessions PHP 100% côté serveur
- Possibilité de gérer des privilèges utilisateurs
- Administration des utilisateurs
- Messages d'erreurs facilement paramétrables
- Séparation stricte code/données/mise en forme
- Code lisible et 100% compatible WYSIWYG grâce à la conservation des blocs HTML

## Sommaire :

# 1- Index.php

A la ligne 4, on débute une session côté serveur grâce à la fonction **session_start()**.

Vous avez du voir que sur cette page il y avait un champ login et un champ mot de passe. Ils sont inclus dans un formulaire (balise html form, ligne 61 : method = POST et action non défini, ce qui revient à poster la page à elle-même).

## *1- Le test*

A la ligne 6, on va donc tester si les variables postées sont définies.

Si c'est le cas, alors on récupère la variable login et mot de passe. (Précisons que les variables POST sont accessibles par $_POST['login'] et $_POST['pass']. (Les noms des variables sont déterminés par le nom donné aux champs de formulaire…).

Les variables $login et $pass vont nous servir pour faire une requête SQL sur notre base de données. Elles sont normalement identiques à leurs variables POST homologues sauf que :
On leur applique la fonction **addslashes()**, qui va permettre d'échapper les caractères spéciaux comme ' qui pourraient être utilisés par des utilisateurs malveillants pour pénétrer notre système en traficotant la requête. Cette technique s'appelle l'injection SQL.
On applique également la fonction **md5()** à notre variable $pass, car notre requête devra faire la comparaison entre le mot de passe tapé par l'utilisateur et l'empreinte md5 du bon mot de passe qui lui se trouve dans notre base de données. (On aurait pu aussi utiliser **sha1()** **ou** **crypt()** (fonctionnement différent)).

Rappel cryptage et chiffrement.

Qui dit cryptage, dit réversibilité. Une chaîne cryptée est normalement décryptable.
Or, dans notre cas, la chaîne mot de passe est chiffré via md5(). C'est-à-dire qu'on fait une empreinte numérique de notre chaîne grâce à un algorithme puissant de hashage qui permet, sur 32 caractères de faire correspondre une chaîne quasi unique et totalement illisible à une chaîne de caractère ou un fichier.
Ainsi, quelqu'un qui disposerait de notre mot de passe chiffré ne pourrait jamais le décrypter avec certitude.

On peut alors faire notre requête "SELECT * FROM utilisateurs WHERE login='$login' AND pass='$pass'" en toute tranquillité.

La fonction **mysql_num_rows()** = va nous permettre de retourner le nombre de ligne du résultat. 1 si un utilisateur est défini comme répondant à la bonne combinaison vraie login et mot de passe. Elle ne renverra rien si la requête ne renvoi pas de résultats. Ce résultat est conservé par la variable $utilisateur.

## *2- La validation*

De là, if ($utilisateur) CAD si la variable utilisateur est définie, alors on va commencer à utiliser toute la puissance des sessions.

Rappel sur les sessions PHP :

Elles permettent de conserver des informations sur l'utilisateur et de maintenir une connexion ayant un identifiant unique entre la machine cliente et le serveur.

On va donc enregistrer toutes les informations utiles par notre interface dans des variables de sessions. Afin de les conserver sans avoir besoin de faire de requête ni de redemander sa combinaison login/mot de passe à notre utilisateur. C'est ce qui se passe de la ligne 23 à 27 : on récupère les valeurs voulues (contenues dans la base) en les déclarant comme des variables ayant pour syntaxe $_SESSION['variable_de_session'].

Une fois ce travail de récupération achevé, on redirige l'utilisateur vers la bonne page auquel il a maintenant droit.

## *3- Le rejet de l'utilisateur*

Regardons tout de même plus loin, pour finir de comprendre le fichier index.php.
La condition de la ligne 18 est suivie de l'instruction else. Que fait-elle ?
Elle va rediriger l'utilisateur en cas d'échec d'authentification vers l'url index.php?erreur=login (à savoir la page actuelle mais avec un paramètre GET en plus).
La présence de cette variable GET ou variable d'url va nous permettre d'afficher un message d'erreur à l'utilisateur. Ces variables se récupèrent comme suit : $_GET['variable'].
Ainsi, regardons à la ligne 63 :
<?php if(isset($_GET['erreur']) && ($_GET['erreur'] == "login")) { // Affiche l'erreur ?>
Cela veut dire que ce qui suit aura lieu si notre url contient erreur=login.

Remarque : Notons ici l'écriture du message d'erreur en html pur …

Dans une optique de séparation totale code/contenu/mise en forme, ce script est développé de manière à ce que l'écriture des instructions PHP permette également l'écriture monobloc de code html. Cela a pour conséquence que toutes ces pages ne contiennent aucune balise cassées et qu'elles sont éditables sous Dreamweaver ou tout autre Wysiwyg. Au final, les graphistes et les personnes chargées de la mise en forme peuvent donc travailler côte à côte avec les développeurs, sans pour autant se marcher sur les pieds… On évitera alors d'avoir à mettre en forme du code avec du html concaténé et coupé (ce qui est rapidement ingérable). Voilà pour l'aparté.

Vous pouvez alors voir que jusqu'à la ligne 70, le fichier index.php gère les différents messages d'erreur que nous serons amenés à rencontrer lors de notre navigation ou nos mauvaises manipulations.

## *4- La déconnexion*

Finalement. On gèrera la déconnexion de l'utilisateur grâce au passage d'une variable GET.
En effet, de n'importe où dans notre site, un lien vers index.php ?erreur=logout engendrera la déconnexion immédiate de l'utilisateur et l'affichage du message d'erreur correspondant.
C'est ce qui se passe à la ligne 38 où nous effectuons le test sur la variable d'url.
On récupère ensuite la variable de session contenant le prénom, afin de dire au revoir à notre utilisateur (soyons polis). Ligne 40 on détruit finalement la session « authentification » en cours, et ligne 41, on redirige une nouvelle fois l'utilisateur pour faire afficher le message

de déconnexion. (On aurait pu tout faire d'un bloc mais on a choisi une gestion des messages d'erreur via l'url).

# 2- Accueil.php

## *1- Protection/Vérification*

Avant toute chose, jetons un œil sur la première partie de ce fichier : ligne 12 à 17. Elle va nous permettre de protéger l'accès à la page d'accueil (privée) en vérifiant si une session authentification est bien enregistrée (if (session_is_registered("authentification"))). **session_start()** va nous permettre ici de relayer automatiquement la session en cours sans l'écraser. Dans le cas contraire, l'utilisateur est redirigé vers la page d'index qui affichera un message d'erreur (scénario d'erreur transmis par l'url).

Maintenant que nous sommes sûr que personne d'autre qu'un utilisateur authentifié ne puisse accéder à notre page, nous pouvons afficher les données confidentielles que nous voulons à notre utilisateur connecté.

## *2 – Affichage des données de session*

Nous avons vu tout à l'heure que la durée de vie des variables de session était égale à la durée de la session elle-même si elles ne sont pas supprimées ou surchargées.
Donc, nous pouvons afficher (comme ce qui est fait) le login, le nom et le prénom de notre utilisateur connecté. C'est ce que vous avez normalement à l'écran.
En regardant la source, vous verrez que <?php echo $_SESSION['variable']; ?> suffit à afficher une variable de session.

De la même façon et en utilisant l'affichage par bloc expliqué plus haut, nous allons pouvoir gérer un affichage conditionnel de blocs html en fonction de certains paramètres contenus dans les sessions. Notre base a en effet été prévue pour contenir un privilège utilisateur. On peut alors facilement imaginer l'intérêt de pouvoir afficher différentes choses selon qu'on soit connecté en tant qu'administrateur ou simple utilisateur.

Donc, par un simple test sur la variable $_SESSION['privilege'], on va pouvoir définir différents scénarios en fonction de cette valeur. Le test a lieu à la ligne 60 et ligne 67.
On affichera alors le bloc correspondant au profil de la session en cours.

Vous êtes normalement connectés avec le login « toto », qui est un administrateur. Il a le droit d'accéder aux fonctionnalités administrateur qui sont ici assez simples : pouvoir créer et supprimer les utilisateurs de la base de données. Je vous laisse donc essayer de créer un utilisateur lambda. Que remarquez vous lorsque vous vous connectez avec ce nouveau profil ? Vous remarquez que sur une même page (accueil.php, on peut faire afficher du contenu différent en fonction de la personne qui est connecté. C'est ça la magie du d'un langage dynamique, c'est ça aussi la magie des sessions.

# 3- Admin.php

## *1- Protection/Vérification*

C'est le fichier uniquement accessible aux utilisateurs ayant des droits administrateurs. Il se compose donc de deux flitres (conditions d'accès) : être authentifié + être administrateur (avoir le privilège administrateur).

Le filtre de protection a donc été agrémenté d'une nouvelle condition (cf. ligne 12)
$_SESSION['privilege'] == "admin"

Si vous me suivez, vous devez donc comprendre que l'on peut aisément créer des pages protégées, même au sein de l'espace protégé, en jouant sur les filtres que sont les groupes utilisateurs.

Le fonctionnement est sinon le même que pour la page accueil.php : si l'utilisateur ne correspond pas à cette condition, il est automatiquement redirigé vers la page index, avec un message d'erreur que l'on défini via la variable d'url « erreur », transmise comme ceci : index.php ?erreur=intru et récupérée comme ça : $_GET['erreur'].

## *2- Gestion des erreurs et des messages de notification*

Idem que dans tout le script : lorsque une condition d'erreur est remplie, une variable d'url est passée à la page en cours ou à la page cible, afin de permettre l'affichage du message sous la condition if(isset($_GET['variable_erreur'])).

(Relire le paragraphe 1-3 + remarque pour plus de détails).

## *3- Ajout d'un utilisateur*

Si toutes les conditions sont remplies :

```
if(isset($_POST['login'])){
        if(($_POST['login'] == "") || ($_POST['pass'] == "")){
                header("Location:admin.php?erreur=empty");
        }
        else if($_POST['pass'] == $_POST['pass2']){
```

Si les variables login et pass de notre formulaire sont définies, si le pass est correctement entré deux fois, alors on exécute la requête de la ligne 33 qui ajoute l'utilisateur (INSERT SQL) à la ligne 35 dans la table utilisateurs.

## *4- Suppression d'un utilisateur*

Pour cette fonctionnalité on utilise un sélecteur liste de formulaire (balise select).

Ce sélecteur va devoir afficher la liste des utilisateurs, puis, lors de l'envoi du formulaire, exécuter la requête permettant de supprimer l'utilisateur choisi.

Pour l'affichage, nous utiliserons les instructions do et **while** (traduction algorithmique : faire, tant que).

>> Tant qu'il y a des utilisateurs dans la base, on affichera leur nom.

La requête sur les utilisateurs est effectuée à la ligne 47 et renseignée à la ligne 46 : "SELECT * FROM utilisateurs ORDER BY nom ASC"

Revenons donc à notre selecteur et observons que le champ option est contenu dans une boucle. Que fait cette boucle ? Pour chaque utilisateur, elle va ajouter un champ option ayant pour valeur l'identifiant de l'utilisateur (ligne 133).

La valeur de ce champs sera une chaîne dynamique concaténée contenant les carractères >> << si l'utilisateur est administrateur, son nom, son prénon, ainsi que son login entre parenthèses.

On pourra donc avoir : >> Turlu TUTU (toto) << ou Turlu TUTU (toto) selon que l'utilisateur soit administrateur ou non

Voici pour l'affichage. Passons maintenant à la suppression à proprement parler…

En fonction de l'utilisateur sélectionné, une fois le formulaire posté, la page recevra la valeur de l'identifiant de l'utilisateur le nom du sélecteur : suppr, deviendra la variable $_POST['suppr'] et la valeur du champ option, la valeur de cette nouvelle variable).

Voici ensuite ce qui est fait à la ligne 50 :

```
if(isset($_POST['suppr'])){ 1- On vérifie la présence de la variable suppr

$id = $_POST['suppr']; 2- On utilise une variable pour collecter cette valeur

$delete_user = sprintf("DELETE FROM utilisateurs WHERE id_user='$id'"); 3- On passe
```

cette valeur à une requête supprimant dans la base l'utilisateur ayant l'identifiant correspondant à l'identifiant passé en variable de formulaire.

L'utilisateur est maintenant supprimé dans la base, il ne suffit plus qu'à le signaler en redirigeant vers la page avec un paramètre d'url :

```
header("Location:admin.php?delete=ok");
```

La variable delete nous permet alors d'effectuer un test sur sa présence afin de pouvoir gérer un contexte de suppression et ainsi d'afficher ou non le message « L'utilisateur a été supprimé avec succès ».

# 4- Connexion.php

Ce fichier est inclus dans toutes les pages par la commande **require_once.**

Il contient tous les variables constants nécessaires à l'exécution des requêtes.

Il est explicitement commenté.

# 5- Lisez_moi.htm

C'est le fichier d'aide. Je vais quand même pas vous à comprendre à quoi sert le fichier d'aide tout de même … ?!
Si vous lisez cette ligne, c'est qu'il ne vous reste plus qu'à acheter **Php et MySQL pour les nuls** …

# 6- Idées d'évolutions

**Evolutions possibles :**

- Empêchement/vérification des logins doublons
- Administration des privilèges (ajout/suppr/modif)
- Possibilité de changement du mot de passe à l'utilisateur connecté
- Inscription en option avec possibilité de modération
- Notifications par email
- Possibilité de choisir la fonction de cryptage à utiliser et d'y ajouter des paramètres
- Installation automatisée
- Création d'une feuille de style paramétrable
- Création de gabarits d'images pour personnalisation (fichier Fireworks+ découpes)
- Travail sur l'accessibilité
- Nettoyer les sources

Ce script a été réalisé et distribué dans un but non lucratif afin de rendre accessible à tous un moyen simple et efficace de sécuriser des pages web. J'espère que vous en ferez bon usage. Si toutefois vous appréciez le service rendu (et comme la gratuité ne nourrit pas son homme), vous pouvez me faire un don via l'interface sécurisée Paypal en cliquant sur le lien suivant.

**FAIRE UN DON**

A votre bon cœur … MERCI !