Objectifs

- ☞ Intéractions PHP/MySQL
- ☞ Gestions des sessions et cookies

Dans ce TP on va ajouter une vraie gestion des utilisateurs et des followers à "Gazouillis", que l'on a commencé la semaine dernière.

Manipulation [Récupérer les fichiers nécessaires au TP, et mettre en place la base de données]

Voici la procédure à suivre :

1. Télécharger `http://pablo.rauzy.name/files/mon/tpweb/tp7.tgz`
2. Décompresser le dans votre dossier "public_html".

Cela vous créera un dossier "gazouillis2" dans votre dossier "public_html".
Ce dossier contient 7 fichiers : index.php, functions.php, user.php, actions.php, register.php, login.php, et db-config.php
C'est le même principe que l'autre fois et c'est à peu près ce que vous avez dû obtenir une fois le TP6 terminé, il y a toutefois des indications en plus et le terrain pour ce TP est un peu préparé, donc je vous conseille de partir de ça plutôt que de votre TP, sauf si vous vous sentez l'âme d'un développeur web.
Ouvrez le fichier db-config.php et entrez y les informations nécessaires.

Exercice 1 [Gestion de mot de passe en base de donnée]

Il est important de savoir que on ne doit pas stocker tels quels les mots de passe de ses utilisateurs. En effet, si vous vous faites attaquer et qu'un "pirate" a accès à votre base de donnée, il ne faut pas qu'il puisse connaître les mots de passe de vos utilisateurs, surtout que ceux ci le réutilise (À TORT) sûrement ailleurs.
Ce qu'il ne faut pas non plus, c'est qu'à partir de ce qui est dans la base de donnée, on puisse retrouver le mot de passe. Donc on ne peut pas le chiffrer puisque si l'attaquant mets aussi la main sur la clef de chiffrement, alors il peut déchiffrer les mots de passe et y avoir accès.
Pour ça, on a inventé des *fonctions de hachaqe*. L'idée, c'est de prendre une chaine (par exemple un mot de passe), et de calculer un nombre qui lui correspond, qui soit unique, et qui soit impossible (c'est à dire très très très dur) à calculer dans l'autre sens.
Par exemple la fonction `sha1` qui est disponible en PHP fait cela :

```
<?php echo sha1('hello'); ?>
```

Va afficher :

```
aaf4c61ddcc5e8a2dabede0f3b482cd9aea9434d
```

Ces 40 caractères correspondent à un nombre hexadécimal qui est un *haché* de la chaine "hello".
Et il est a priori impossible de calculer "hello" à partir de "aaf4c61ddcc5e8a2dabede0f3b482cd9aea9434d". Sauf que c'est très facile de le faire dans l'autre sens, alors il y a des gens qui ont prévu le coup et on calculé les `sha1` de tous les mots possible de moins de 10 lettres et de tous ceux du dictionnaire par exemple. Et je suis sûr que du coup en le cherchant bêtement sur Google on retrouve que ça correspond à "hello".
Alors l'astuce c'est de mettre ce qu'on appelle du *sel*. Pour que les hachés soient sécurisé, on rajoute dans la chaîne un autre texte qui fait que quelque soit ce que l'utilisateur utilise comme mot de passe, le haché de son mot de passe + le sel n'aura pas été calculé à l'avance.
Par exemple on pourrait utiliser le sel "pouet pouet gazouillis lalala 42 MTX AGRAL" en se faisant une fonction comme :

```php
<?php
function password_hash ($password)
{
  return sha1('pouet␣pouet␣gazouillis␣lalala␣42␣MTX␣AGRAL'.$password);
}
?>
```

1. Copiez le contenu de la fonction dans functions.php à la ligne 5.

Maintenant on doit ajouter un champ pour stoquer les hachés des passwords dans la base de donnée.

2. Ajouter un champ “password” dans la table “gzl_users”, il doit être après le champ “email” et être de type `varchar(40)` et non NULL (vous pouvez faire cela dans PHPMyAdmin ou directement en SQL, au choix).
3. Mettre pour à jour le password des utilisateurs existants avec le haché du mot de passe “hello” (calculé avec la fonction `password_hash` qu’on a défini juste avant) : “9a88ad51643f791abf0cd7f2c42f92f63dfaa756”.

Exercice 2 [Mise à jour du formulaire d’inscription]

Il faut maintenant demander un mot de passe à l’utilisateur quand il s’inscrit.

1. Dans register.php, rajouter un champ de type “password” dans le formulaire et qui a pour nom “password”.
2. Dans actions.php, rajouter un argument à la fonction `create_user`, en dernière position et qui correspond au champ password que le formulaire de register.php a envoyé.
3. dans functions.php, mettre à jour la fonction `create_user` pour qu’elle prenne l’argument supplémentaire et ajoute le mot de passe dans la table “gzl_users” :
   (a) ajouter un argument `$password` à la fonction.
   (b) hacher le password avec notre fonction `password_hash`.
   (c) ajouter le password dans la requête SQL pour qu’il soit ajouter dans la base de donnée.

Exercice 3 [Permettre à un utilisateur de se connecter]

Il faut maintenant permettre aux utilisateurs de se connecter. Il faut pour cela créer le formulaire de login et leur permettre d’ouvrir une session.

1. Dans index.php, user.php, et actions.php, tout en faut du fichier, dire qu’on utilise les sessions en appelant la fonction `session_start`.
2. Dans login.php :
   (a) Créer un formulaire qui utilise la méthode POST et qui a pour action `actions.php?action=login`.
   (b) Il doit comporter deux champs texte : “pseudo”, et “password”, ainsi qu’un bouton submit.
3. Dans functions.php, écrire la fonctions `login_user`. Cette fonction reçoit un pseudo et un mot de passe en arguments.
   (a) Récupérer une version hashé du password avec la fonction `password_hash`.
   (b) Récupérer dans la base de donnée l’id de l’utilisateur qui correspond à ce pseudo et ce mot de passe. Si il n’y en a pas, alors retourner une erreur “mauvais utilisateur et/ou mot de passe”.
   (c) Si on a bien récupéré l’id de l’utilisateur, on veut le mettre dans la variable globale `$_SESSION['id']` et on retourne `null`.

Exercice 4 [Gérer les utilisateurs connectés]

On va maintenant surveiller si un utilisateur est connecté et afficher des choses différentes si c’est le cas.

1. Dans index.php chercher les mentions de `À MODIFIER - exo 4` :
   (a) Pour les trois premières, remplacer les `false` dans les conditions par un test qui vérifie qu’un utilisateur s’est bien connecté. Ici on va vérifier que $_SESSION[’id’] existe (avec la fonction `isset`).
   (b) Pour la dernière, on veut vérifier que c’est bien l’utilisateur connecté qui a écrit le gazouilli pour lui donner le lien qui pemret de le supprimer, il faut donc non seulement vérifier si $_SESSION[’id’] existe, mais aussi si il correspond à l’`user_id` du gazouilli.

2. Dans user.php chercher les mentions de `À MODIFIER - exo 4:`
   (a) Pour la première, on veut vérifier si l'utilisateur connecté et si oui si c'est celui dont on est en train de regarder le profil.
   (b) Pour l'autre, c'est comme pour la dernière de index.php.
3. Dans functions.php :
   (a) Dans la fonction `create_gazouilli` on veut que le `user_id` soit celui de l'utilisateur connecté.
   (b) Dans la fonction `delete_gazouilli` on veut être sûr que le gazouilli que l'on supprime a été créé par l'utilisateur connecté.
   (c) Dans la fonction `delete_user` on veut être sûr que l'utilisateur que l'on supprime est l'utilisateur connecté, si ce n'est pas le cas, retourner l'erreur "Vous n'avez pas le droit de supprimer un autre utilisateur".

Exercice 5 [Déconnexion d'un utilisateur]

On veut maintenant autoriser un utilisateur à se déconnecter. Dans functions.php dans la fonction `logout_user` :

1. Détruire la session en utilisant la fonction `unset` pour détruire les variable qu'on a mis dedans.
2. Ensuite appeler la fonction `session_destroy` puis `session_write_close`.
3. Retourner `null` pour signaler que ça s'est bien passé.

Exercice 6 [Rester connecté même quand on quitte le navigateur]

Si vous vous connecté sur Gazouillis dans l'état actuel des choses, vous le resterai aussi longtemps que votre navigateur reste ouvert, mais si vous le fermez et que vous revenez sur le site, vous devez vous relogguer.

La solution pour remédier à cela est d'utiliser les cookies afin de donner la possibiliter la l'utilisateur de rester connecter pour un certain nombre de jours au moment où il se connecte.

1. Dans login.php, rajouter une checkbox que l'utilisateur cochera si il souhaite rester connecté. Mettez lui "remember_me" comme nom et "yes" comme valeur.
2. Dans actions.php, modifier l'appelle à la fonction `login_user` pour ajouter le nouveau paramètre `$_POST['remember_me']`.
3. Dans functions.php, modifier la fonction `login_user` pour qu'elle prenne en compte le nouveau paramètre `$remember_me` puis :
   (a) Avant le `return null;` si tout s'est bien passé, rajouter un test pour voir si le nouvel argument `$remember_me` vaut bien "yes".
   (b) Si oui, alors mettre un cookie qui s'appelle "gazouillis" qui contient l'id de l'utilisateur et qui expire dans une semaine avec la fonction `setcookie`. Normalement on ne mettrait bien sûr pas directement l'id de l'utilisateur puisque c'est bien trop simple à forger pour se faire passer pour un autre, mais ici on veut faire simple.
4. Ensuite dans index.php, on regarde si le cookie existe (il est dans la variable globale $_COOKIE['gazouillis'], utiliser `isset` pour vérifier) et si oui, on appelle la fonction `cookie_login_user`. Faire la même chose dans user.php.
5. Dans functions.php, remplir la fonction `cookie_login_user` pour mettre l'id de l'utilisateur dans `$_SESSION['id']` comme dans `login_user` quand tout s'est bien passé.
6. Dans functions.php, supprimer le cookie en le rendant invalide tout de suite dans la fonction `logout_user`.

Exercice 7 [BONUS : Gestion des followers]

1. Créer une table de jointure follower-followed avec deux champs de type entiers et not null, qui seront les id des deux utilisateurs.
2. Sur la page d'un utilisateur (dans user.php), quand on est loggé et qu'on est pas l'utilisateur en question, rajouter un lien vers `actions.php?action=follow&id=ID` ou `ID` est celui de l'utilisateur dont on voit la page.

3. Dans functions.php, coder le contenu de la fonction `follow_user` pour qu'elle rajoute ce qu'il faut dans la table que vous avez créer à la question 1.
4. Dans functions.php, adapater la reqête SQL de la fonction `fetch_user_timeline` pour qu'elle récpère seulement les gazouillis de l'utilisateur connecté et des utilisateurs qu'il follow.
5. Dans functions.php, écrire une fonction qui étant donnée un id d'utilisateur, dit si l'utilisateur actuellement connecté le follow ou non.
6. Utiliser cette fonction pour afficher un lien vers `actions.php?action=unfollow&id=ID` plutôt que comme dans la question 2 en fonction de son résultat sur la page d'un utilisateur quand celui qui la visite est connecté.
7. Écrire la fonction `unfollow_user`.