Formations en Informatique de Lille – 2014/2015

Semaine 10 - Technologies du Web

mars 2015

Authentification, upload, images et base de données

Exercice 1 : Vous trouverez un corrigé de l'exercice réalisé en semaine 6 (authentification). Ce corrigé sera le point de départ de l'exercice d'aujourd'hui.
Pour rappel, il s'agissait de construire un système d'authentification qui reposait sur le principe suivant

1. Chaque utilisateur enregistré dans l'application possède un **identifiant** (ici il s'agit de son nom de login). Les identifiants et mots de passe figurent sur le serveur en mémoire permanente (fichier texte bientôt remplacé par une table de base de données).
2. On choisit et maintient un **témoin** pour indiquer si l'utilisateur est authentifié ou non. Ce témoin est une clé du tableau `$_SESSION` (ici il s'agit de `$_SESSION['ident']`. Cette variable est définie si et seulement si l'utilisateur est authentifié. Dans ce cas, sa valeur est l'identifiant de l'utilisateur.

Le système repose sur les composants suivants

1. La classe `Personne` représente les informations de base sur un utilisateur enregistré (login, nom, prénom)
2. La fonction `authentifier($login, $password)`. vérifie la validité du login et de son mot de passe. S'ils sont corrects elle renvoie une instance de `Personne` correspondant au login fourni. Sinon elle renvoie `null`
3. La fonction `controleAuthentification()` chargée de vérifier ou de procéder à l'authentification de l'utilisateur. En cas de nouvelle authentification, elle provoque la restauration du contexte utilisateur.
   Cette fonction déclenche une exception si et seulement si l'authentification a échoué
4. Le script `tpauth` est une **sentinelle** : Si l'utilisateur est authentifié, il ne fait rien. Si l'utilisateur n'est pas authentifié, il inclut une page par défaut (ici le formulaire de login) et **provoque la terminaison du script**. Le script `tpauth` est destiné à être inclu en en-tête d'une page que l'on veut réserver aux utilisateurs authentifiés.

Question 1.1 : Le but de cette question est de mémoriser les informations utilisateurs dans une table de la base Postgres, en remplacement du fichier password.txt.
Construire tout d'abord une table destinée à remplacer le fichier "password" : elle comportera des attributs login, password, nom et prenom. Vous ferez en sorte que l'attribut password puisse contenir des chaînes allant jusqu'à 200 caractères (la raison apparaîtra à la question suivante).
Modifier la fonction `authentifier` en conséquence.

Les 2 versions que nous venons de voir souffrent d'un grave problème : les mots de passe des utilisateurs sont stockés «en clair». Une base d'utilisateurs ne devrait **jamais** procéder ainsi. Tout d'abord l'administrateur de site n'a pas à connaître les mots de passe des usagers (c'est une donnée privée et souvent les utilisateurs emploient le même mot de passe pour plusieurs sites ou applications). Ensuite, en cas de vol de données, tous les mots de passe utilisateurs peuvent de retrouver divulgués.

La technique à employer est celle du hachage : on applique au mot de passe une fonction de hachage h. Si le mot de passe est m on ne mémorise pas m mais $z = h(m)$. À chaque connexion, on applique la fonction h au mot de passe fourni et on compare le résultat avec z.
Le choix de la fonction h est bien évidemment crucial. Ce doit être une fonction «à sens unique», c'est à dire que, même en connaissant z, il doit être «impossible» (du moins dans un délai de temps à l'échelle humaine) de retrouver m ni une autre chaîne s telle que $h(s) = z$.
La fonction `crypt()` implémente plusieurs algorithmes de hachage cryptographique. Le premier argument est la chaîne à hacher. Le deuxième est optionnel mais recommandé : il s'agit du «salage». Dans le cas de la fonction crypt la forme du salage détermine aussi l'algorithme à utiliser.
L'un des algorithmes les plus sûrs est BLOWFISH. Pour que crypt l'utilise, il faut un salage sous la forme `$2a$`*nn*`$` suivi de *22 lettres ou chiffres quelconques*, soit une chaîne de longueur 29.
Si la chaîne fournie est plus longue, seuls les 29 premier caractères (dans le cas de Blowfish) sont pris en compte.
nn est une valeur décimale à 2 chiffres comprise entre 4 et 31 indiquant un nombre d'itération de l'algorithme (une valeur inférieure ou égale à 10 suffira).
Pour une meilleure sécurité, les 22 caractères sont choisis "au hasard" pour chaque mot de passe à hacher.
Le résultat de crypt est une chaîne qui commence par le salage utilisé, suivi du résultat du hachage lui-même. Il convient d'en mémoriser la totalité, car il faudra réutiliser le même salage à chaque fois que l'utilisateur fournit son mot de passe.
Le résultat de crypt peut être long, d'où le dimensionnement choisi pour l'attribut `password`.
En résumé, on procèdera de la façon suivante :

1. Lors de la création du mot de passe, choisir de façon pseudo aléatoires 22 lettres ou chiffres de manière à constituer un salage *salt* de 29 caractères puis calculer $crypt(m, salt)$ et le mémoriser dans l'attribut password.
2. Quand un utilisateur se connecte en fournissant un mot de passe m : retrouver la valeur mémorisée dans l'attribut password (on l'appellera z) et vérifier si $crypt(m, z) == z$

Question 1.2 : Remplacez, dans la table, les mots de passe par leur valeur hachée.

`animal`	`$2a$10$bEDyUE4/PJn.8MXO.8Hzbe8JQOI37tM8XlxHkiudhkyW2PcHqr1oa`
`mystere`	`$2a$10$CEONn8abicGMBoftnU3BSuld2PlRwaKSB5i6qlb/Ym/oYCsINdFxK`

Puis modifiez de nouveau la fonction `authentifier` en conséquence.

Question 1.3 : Créez une page d'enregistrement d'un nouvel utilisateur : le formulaire demandera les 4 informations nécessaires (login, password, nom et prénom).

Question 1.4 : Créez une page «mon compte» qui permet à un utilisateur enregistré de modifier les informations le concernant. Il peut modifier son mot de passe mais pas son login.

Question 1.5 : On souhaite qu'un utilisateur puisse enregistrer ses centres d'intérêts. Un centre d'intérêt sera désigné par une chaîne (limitée à 20 caractères maxi et dans laquelle la virgule est interdite).

— Construire une table `interets` à 3 attributs : login, sujet, index. L'attribut index sera un entier constituant une clé primaire. Vous ferez en sorte qu'il soit généré par incrémentation automatique.
— Ajouter à la page «mon compte» un champ de formulaire permettant de préciser une liste de centres d'intérêts séparés par des virgules.

Question 1.6 : Construire un « web service » `find_user.php`. qui **peut** recevoir en mode GET un paramètre `interest` désignant UN centre d'intérêt. Le web service renvoie la liste des utilisateurs enregistrés. Si `interest` est fourni, seuls les utilisateurs ayant ce centre d'intérêt seront listés.
La réponse sera fournie en JSON sous la forme d'un tableau d'utilisateurs et pour chacun d'eux son login, son nom et son prénom.
Par exemple

```
[
   {"login":"mallani","nom":"Malle","prenom":"Annie"},
   {"login":"toto","nom":"Toto","prenom":"Alfred"}
]
```

Question 1.7 : Vous trouverez ici les informations techniques sur l'upload de fichier ainsi que sur la façon de ranger des données binaires dans une BD via PDO
Ajoutez la possibilité pour l'utilisateur enregistré de définir un avatar (il l'enverra par upload) et modifiez la page web pour afficher cet avatar.

Question 1.8 : Quand une image n'est destinée qu'à être affichée à taille réduite, la mémoriser en grande taille est un inconvénient. D'abord pour la taille de la base de données mais encore bien d'avantage pour le temps de chargement de la page web : le navigateur va devoir charger un fichier inutilement volumineux.
Un avatar est destiné à n'être affiché qu'en taille 70 maximum (plus grande dimension). Il est de préférence carré.
Vous allez modifier l'image envoyée par l'utilisateur (uniquement si l'une de ses dimensions est supérieure à 70px) :

— ne retenir que le plus grand carré possible, au centre de l'image
— réduire ce carré à une image 70x70 ;

Vous utiliserez pour ce faire les fonctions PHP : `imagecreatefromjpeg` (et/ou `imagecreatefrompng`, `imagecreatefromgif`,`imagecreatefrombmp`) `imagecreatetruecolor`, `imagecopyresampled` et `imagepng`
En PHP pour manipuler une image il faut d'aord construire sa représentation en mémoire (i.e. qui peut être rangée dans une variable)

— création à partir d'un fichier :
`$image = imagecreatefromjpeg($nomFichier)`
pour les fichiers JPEG. Il existe des fonctions équivalentes pour les autres formats.
— création d'une image noire (qui servira de base à des manipulations) :
`$imageNouvelle = imagecreatetruecolor($largeur,$hauteur)`

On peut copier une image dans une autre, en la transformant éventuellement

— `imagecopyresampled($imageNouvelle,$image,....)`
cette fonction permet de créer une copie de l'image d'origine tout en coupant et/ou redimensionnant

On peut ensuite sauver l'image dans un fichier, dans le format de son choix (ici : PNG)

— `imagepng($imageNouvelle,$nomFichierDestination)`