TP4 - PHP/MySQL

Exercice N°1 :

Il s'agit de faire envoyer des messages lors de l'exécution de scripts PHP.

1. Le message peut être envoyé automatiquement en réaction à un évènement rencontré lors de l'exécution, par exemple une erreur interceptée, une mise à jour affectée par un utilisateur dans une base de données, …
   Ou bien, ce qui est l'exemple traité ici, le message peut être envoyé par un utilisateur lors de la validation d'un formulaire qu'il aura dûment renseigné.
   Si l'envoi d'un message textuel ne pose pas de problème majeur (si le paramétrage du PHP le permet bien sûr !), en revanche l'expédition de pièces jointes est bien plus délicate, on ne traitera pas ça dans ce TP .

2. L'envoi du message quel qu'il soit, sera effectué par la fonction **mail()**
   (référence : *http://www.php.net/manual/fr/function.mail.php*)
   Elle s'efforce de poster un message à une liste de destinataires et renvoie un booléen de compte-rendu.

3. Syntaxe

   `mail ($liste, $sujet, $message, $entete)` où :

```
$destinataire : liste de  destinaires du message
$sujet : sujet
$message : le corps du message
$entete : l'adresse de l'expéditeur,  et éventuellement l'adresse
reply-to.
  par exemple : "From: $adresse_expediteur\r\nreplyTo:
$adresse_expediteur"
```

4. Problèmes rencontrés

   Attention une valeur TRUE renvoyée indique seulement que la fonction *mail()* a correctement transmis le message au système sous-jacent, mais n'est pas un indice de bon acheminement, ce n'est ni un reçu, ni un accusion de réception ! (rép. http://www.php.net/manual/fr/ref.mail.php)
   Si on rencontre des problèmes, d'abord bien vérifier que PHP trouve un agent d'expédition (sendmail, postfix ..) sous Linux ou que le paramétrage de *php.ini* est bien correcte (sous Windows)
   Configuration sous Easyphp Si on teste ses scripts avec la plate-forme de test **easyphp** sous Windows, il est indispensable d'intervenir dans la configuration de PHP, déterminée par le fichier **php.ini** situé dans le sous-rép. *easyphp\apache* (pour Easyphp 1.6 situé dans *C:\windows\php.ini*)
   Renseigner les 2 lignes ci-dessous avec le smtp du fournisseur d'accès et votre adresse de retour.

```
[mail function];
;;;;;;;;;;;;;;;;;;;
; Module Settings ;
;;;;;;;;;;;;;;;;;;;
;For Win32 only.
SMTP = smtp.xyz.fr
;For Win32 only.
sendmail_from = xxx.yyy@zzz.fr
```

Exercice N°2:

Il s'agit de mettre en place un système d'authentification (basé sur la notion de *session*) permettant de restreindre l'accès d'un site Web (ou d'une partie du site) à un ensemble d'utilisateurs identifiés et appartenant à une liste établie. La liste des utilisateurs est stockée dans une table (***Membres) d'une base de données MySql.*** Cette table possède les attributs suivants :

- *id* *int (auto_increment - clé primaire)*
- *pseudo* *varchar(20)*
- *motpasse* *varchar(20)*

Notons que la table Membres est à remplir manuellement

Si l'utilisateur est identifié alors on lui donne le droit d'accès à une page privée où il pourra envoyer des mails. Réutilisez le résultat de l'exercice N°1.

On propose de décomposer le système en trois *sous-modules* :

- le formulaire d'authentification (*authentification.htm*)
- le programme traitant le formulaire (*authentification.php*)
- le programme de vérification d'accès à une ressource (*autorisation.php*).

Question 1 - authentification.htm
Ecrire en *HTML* le formulaire d'authentification. Il contient deux zones de saisie, permettant à l'utilisateur de donner son pseudo et son mot de passe.

Question 2. – *authentification*.php.
Le programme *authentification*.php utilise une fonction *verif()* qui vérifie dans la base de donnée si le pseudo et le mot de passe de l'utilisateur existent. Elle renvoie *True* ou *False* selon le résultat et à définir dans un fichier externe.

Question 3. – autorisation.php
Pour vérifier si l'utilisateur est déjà authentifié, nous allons lire son pseudo dans la session. En d'autres termes, si l'utilisateur est identifié, la variable de session *$_SESSION*[*'pseudo'*] existe.