IUT de Montpellier - Semaine Web - TD4
Introduction à PHP - Les sessions

Semaine du 5 Novembre 2012

Le site web de votre entreprise (comme beaucoup d'applications web d'ailleurs) est architecturé en deux parties distinctes : Le Front Office (sur lequel vous avez travaillé actuellement) est la partie destinée au public, tandis que le Back Office d'une application web est la partie dédiée à l'administration. Seuls certains utilisateurs enregistrés, communément appelés *administrateurs*, peuvent y accéder.

1 Enregistrement des utilisateurs

Créez un formulaire pour d'inscription, qui enregistre un utilisateur dans une table *users*.

Attention, il n'est pas pertinent de stocker en clair, les mots de passe des utilisateurs, utilisez plutôt md5 :

```
<?php
$mot_passe_en_clair = 'apple';
$mot_passe_crypte = md5($mot_passe_en_clair);
echo $mot_passe_crypte;  //affiche '1f3870be274f6c49b3e31a0c6728957f')
?>
```

2 Accès aux pages réservées

Pour accéder à une page réservée, un utilisateur doit s'authentifier. Une fois authentifié, un utilisateur peut accéder à toutes les pages réservées sans avoir à retaper son mot de passe.

Il faut donc faire circuler l'information "s'être loggé" de pages en pages. C'est possible à l'aide d'un formulaire caché, mais est ce pertinent ?

2.1 Les sessions

Une session permet d'associer à une navigation (même adresse ip, même navigateur,...) un ensemble de valeurs définies de manière transparente pour le visiteur et que le serveur conserve de page en page. Le maniement des valeurs de sessions est assez simple en PHP, on peut stocker presque tout dans une variable de session : un chiffre, un texte, voire un objet (en passant par une "serialisation").

Dans toute page qui manipule les sessions `session_start();`

Mettre une variable en session `$_SESSION['login'] = 'remi';`

Vérifier qu'une variable existe en session `if ((!empty($_SESSION['login']))) {//do something}`

2.2 Sécurisation des pages à accès réservé

authentification.php est la page appelée par le formulaire d'authentification

```
<?php
session_start(); //premiere instruction de toute page manipulant une session
/*
 * Récupération des login/passwd en POST
 * Vérification dans la base
 * remplissage de la variable $is_ok
 */
$is_ok = true;

if ($is_ok) {
$_SESSION['login'] = 'remi';
echo "Vous etes authentifie";
}
?>
```

une_page_reserve.php Une page a accès restreint aux seuls utilisateurs authentifiés.

```
<?php
require 'est_autorise.php';  //require et pas include

echo "Vous etes sur une page reserve</br>";
$login = $_SESSION['login'];
echo "Bonjour $login";
?>
```

est_autorise.php Le script qui est requis par toutes les pages a accès réservé.

```
<?php
session_start();
if ((isset($_SESSION['login'])) && (!empty($_SESSION['login']))) {
//do nothing
}
else {
// pas de login en session : redirection sur la page public
header('Location: page_public.php');
}
?>
```

3 Enregistrement avec une adresse email valide

Dans le TD 2, vous avez vu comment vérifier qu'un champs texte a le format d'une adresse email valide. Mais vous n'avez pas garanti que cette adresse email existe réellement, ni que son propriétaire est bien la personne en train de s'enregistrer sur votre site.

Pour vérifier que l'utilisateur utilise sa propre adresse email. Générez un identifiant unique avec la fonction uniqid() (`http://php.net/manual/fr/function.uniqid.php`) que vous stocker en base, puis envoyez lui un mail contenant un lien de validation (utilisant cette chaine).

Un utilisateur n'est autorisé à s'authentifier, que s'il a reçu le mail et cliqué sur le lien de validation.

4 Sessions vs. Cookies

Un cookie est utilisé pour stocker une information spécifique sur l'utilisateur, comme les préférences d'un site ou le contenu d'un panier d'achat électronique. Le cookie est un fichier qui est stocké directement sur la machine de l'utilisateur, il s'agit d'une association nom/valeur. Il ne faut pas stocker de données critiques dans les cookies !

Utilisations en PHP :

- La ligne ci-dessous créée un cookie nommé "TestCookie" contenant la valeur $value et qui expire dans 1h.
  `setcookie("TestCookie", $value, time()+3600);  /* expire dans 1 heure */`
  Attention, comme les session ou la fonction header(), la fonction setcookie() doit être appelée avant tout utilisation de HTML (le protocole HTTP impose cette restriction).
- Accéder à un cookie :
  `echo $_COOKIE["TestCookie"];`
- Effacer un cookie (il suffit de le faire expirer) :
  `setcookie ("TestCookie", "", time() - 1);`

5 Autres sécurisations :

Le fait de crypter les mots de passe (ou les numéros de carte de crédit) dans la base de données évite qu'un accès en lecture à la base (suite à une faille de sécurité) ne permettent à l'attaquant de récupérer toutes les données de tous les utilisateurs.

On peut aussi crypter le mot de passe sur le navigateur. Dans ce cas une attaque du tiers-écouteur (type **Man in thé midle**) ne permet pas d'obtenir le mot de passe en clair de l'utilisateur. Mais puisque l'authentification repose sur le mot passe crypté, le tiers peut s'authentifier avec le mot de passe crypté, qu'il a récupéré.

La seule façon fiable de sécuriser une application web est le recours au cryptage de l'ensemble des communications entre le client (browser) et le serveur, via l'utilisation du protocole `ssl` sur `http`, à savoir `https`.

6 Votre site web :

Votre site web contient des produits, dont la description est stoquée dans une base de donnée. Créez un back-office, avec identification, permettant d'ajouter/modifier/supprimer des produits dans la base.