Module S2301- Semaine bloquée Programmation Web - Projet "Site de veille d'information personnalisé"

Partie 3 : version multi-utilisateurs

1. Tâche

Cette partie a pour objectif de permettre l'utilisation du site par plusieurs utilisateurs, chacun ayant défini ses propres flux et éventuellement ses catégories de flux.

2. Sessions

Le protocole http est un protocole sans état. Cela signifie que par défaut, le serveur web ne peut, par exemple, pas savoir que deux requêtes http successives émanent du même internaute. Les sessions permettent de gérer ce problème. Elles s'appuient en général sur un cookie identifiant déposé sur l'ordinateur de l'internaute. A chaque fois qu'une requête est envoyée par l'internaute, le cookie déposé au premier échange http est récupéré par le serveur. Ce cookie lui permet de savoir de qui il s'agit et donc de récupérer l'historique de la communication avec cet utilisateur stocké côté serveur. En PHP, l'historique est représenté par un tableau appelé $_SESSION dans lequel sont stockées les informations propres à un utilisateur, par exemple, son login, ses commandes (dans le cas d'un site commercial), ses préférences etc... Pour accéder au tableau $_SESSION dans un script PHP, il faut préalablement appeler la procédure session_start();.

Dans le cadre de ce projet, nous allons utiliser les sessions pour gérer plusieurs utilisateurs. En effet, considérons le scénario suivant :

1. Un utilisateur se logue (login + mot de passe) Client --> Serveur
2. Les nouvelles concernant ses propres flux sont affichées Serveur --> Client
3. L'utilisateur demande un nouvel affichage Client --> Serveur

Comment savoir de qui provient cette demande d'affichage ? C'est important de le savoir car tous les utilisateurs ne sont pas abonnés aux mêmes flux. C'est là que les sessions interviennent. Au moment du log (1), le script côté serveur vérifie que l'utilisateur est référencé (comparaison du login et mot de passe reçu avec les valeurs stockées dans la base de données), puis (si l'utilisateur est reconnu) ajoute le login à la session, par exemple avec l'instruction suivante : `$_SESSION['login']=$_POST['login'];` Il retourne ensuite la page des nouvelles (2). A la nouvelle demande d'affichage (3), grâce à `$_SESSION['login']`, le serveur peut vérifier que l'utilisateur est logué et savoir de qui il s'agit pour lui renvoyer les nouvelles qui le concernent.

Vous trouverez des explications supplémentaires et des informations utiles sur :

- https://www.php.net/manual/fr/book.session.php
- http://www.lephpfacile.com/cours/18-les-sessions
- http://php.developpez.com/cours/sessions/?page=page_2#LII-A

3. Page de login

Créer d'abord une page de login (***index.html***). Vous utiliserez un champ login (input type text) et un champ mot de passe (input type password) dans un formulaire avec une méthode de type post (mieux vaut ne pas passer la valeur du mot de passe dans l'url !!). Créer un utilisateur dans la base de données. Créer un script ***login.php*** vérifiant login et mot de passe (par comparaison de la valeur des paramètres avec le contenu de la base de données). Dans le cas où l'utilisateur est reconnu, on fera une redirection vers la page d'affichage des nouvelles. Dans le cas contraire, on redirigera vers la page ***index.html***. Rappel, en PHP, pour rediriger vers une autre page, on utilise l'instruction suivante : `header('location:autrepage');`

4. Contrôle des accès aux pages

On va maintenant modifier les pages développées jusqu'à maintenant pour intégrer l'aspect multi-utilisateurs. Chaque page commencera donc maintenant par la vérification de l'existence de $_SESSION['login']. En cas d'échec, on redirigera vers la page ***index.html*** sinon on tiendra compte de la valeur de $_SESSION['login'] dans les requêtes à la base de données pour ne sélectionner que les flux et catégories propres à l'utilisateur.

5. Facultatif

Vous pouvez aussi :

- Créer une page permettant à un utilisateur de changer de mot de passe
- Créer un utilisateur particulier de type administrateur qui aura accès à une page lui permettant de gérer les utilisateurs (ajout, suppression, changement des mots de passe)
- Crypter les mots de passe (en utilisant `password_hash` et `password_verify`, lire : https://www.php.net/manual/fr/book.password.php)