Programmation Web 2

13. Gestion de l'état, sécurité et variables sessions

Le concept d'état

- Il est très souvent crucial, dans une application, de connaître l'état de l'application. Par exemple, chaque opération que vous faites dans Word change l'état du programme. Si vous cliquez sur le bouton Italique, l'état du programme change, et vous écrirez désormais en italique, jusqu'à ce que vous changiez à nouveau l'état du programme.
- Le concept d'état est toujours un problème dans les applications Web. En effet, le protocole HTTP, à la base, est un protocole sans état. La différence est donc énorme avec les applications Desktop (variables locales en mémoire vs. échange d'informations).
- Pourtant, il est souvent important pour une application Web de savoir dans quel état est l'échange en cours avec le client pour déterminer les possibilités suivantes. Par exemple, il est important de savoir si un usager est authentifié avant de lui donner accès à certaines pages du site et il est même important de connaître l'ID de cet usager dans le cas de pages dynamiques qui varient en fonction de l'identité de l'usager.

Solutions

- Il est possible, par exemple, de mettre l'information que l'on veut réutiliser à chaque transaction dans des champs cachés.
  - Pénible, répétitif et pas vraiment sécuritaire.
- Il est possible de toujours répéter l'information dans les query strings de vos URLs.
  - www.test.com/pageprotegee?log=oui ou encore www.test.com/pageprotegee?user=test&password=test
  - Évidemment… pas très sécuritaire.
- On peut faire des requêtes à une base de données pour y stocker et y reprendre de l'information persistante.
  - Peu performant, et pas toujours pratique de faire des BDs pour des renseignements peu importants.
- Deux solutions sont réellement envisageables (et compatibles entre elles) : les cookies et les variables sessions

Témoins (cookies)

- Les cookies sont des petits fichiers stockés sur le disque dur du côté *client*. Ceux-ci sont renvoyés au serveur (et les informations stockées par PHP dans les variables $_COOKIE[ ]) avec toutes les requêtes envoyées par le client, ce qui permet en effet de créer un état entre les requêtes.
- Les cookies ont une mauvaise réputation (qu'ils ne méritent pas), et certains usagers les refusent dans leur fureteur.
- Le défaut des cookies est qu'ils sont sur le côté client et que les usagers peuvent donc les modifier. De plus, comme ils ne sont pas sur le serveur, vous n'avez pas le contrôle sur la sécurité de leur environnement.
- Comme ils transitent sur le réseau, il est préférable d'encrypter l'information qui s'y trouve si elle est de nature personnelle.
- Généralement, les cookies sont utilisés pour sauvegarder les préférences d'un utilisateur.

Variables sessions

- Les variables sessions, au fond, ne sont qu'une combinaison de deux concepts.
  - Un espace de données contenant des variables sur le serveur, qui contient toute l'information et qui peut donc être gros, puisqu'il ne transite pas sur le réseau
  - Un cookie sur le client, qui contient seulement un identifiant appelé *identifiant de session* qui référe à l'espace de données sur le serveur.
- Les sessions se ferment lorsque l'usager ferme son fureteur.
- Il est aussi possible d'utiliser les sessions sans utiliser les cookies, si jamais le fureteur du client ne permet pas l'utilisation des cookies. Il faut cependant en prendre compte lors de la programmation si l'on veut permettre l'utilisation de sessions sans cookies. Nous verrons comment plus tard.

Cookies vs. Variables sessions

- Durée de vie :
  - Vous pouvez spécifier une durée de vie aux cookies, ce qui fait que l'information peut être réutilisée ultérieurement, plusieurs jours après le dernier accès au site.
  - Les variables sessions sont détruites après chaque fin de session. L'information est donc perdue assez rapidement.
- Taille :
  - Les fureteurs limitent la taille des cookies, puisqu'ils sont transmis à chaque requête.
  - Les variables sessions sont stockées du côté serveur, et ont donc une taille illimitée que vous pouvez gérer.
- Sécurité :
  - Les cookies peuvent être modifiés par l'usager. Si vous stockez de l'information importante, comme par exemple un panier de magasinage, il pourra être modifié directement par l'usager, ce qui est évidemment indésirable.
  - Les variables sessions sont à l'abri de l'usager, du côté serveur. De plus, les variables sessions ne transitent pas sur le réseau. Seulement l'identifiant de session transite.

Utilisations des cookies

- Pour définir un cookie, il suffit d'utiliser la fonction setcookie, qui prend en paramètres obligatoires le nom du cookie, sa valeur et un *timestamp* UNIX qui définit le temps exact où le cookie expirera. Il prend aussi en paramètres facultatifs le chemin qui contient les pages pour lesquelles le cookie est important, le domaine pour lequel le cookie est important (pour un site ayant plusieurs domaines) et un entier nommé *secure*, qui permet de spécifier que le cookie ne sera envoyé que si l'encryption SSL est utilisée pour la page.
  - setcookie(“couleur”, “FFFFFF”, time() + 3600);
  - Ce cookie, nommé couleur, aura la valeur FFFFFF et expirera dans une heure.
  - setcookie(“couleur”, “FFFFFF”, time() + 3600, “/mestrucs/pages”, “.monsite.com”, 1)”;
  - Ce cookie ne sera valide que dans les sous-répertoires et les pages contenus dans /mestrucs/pages sur votre site. De plus, il sera valide pour tous les domaines de monsite.com (par exemple images.monsite.com).
- *Cet appel à setcookie() doit être fait avant tout HTML.*

Utilisations des cookies (suite)

- Il est très simple d'obtenir les valeurs des cookies. En effet, PHP remplit la superglobale $_COOKIE[] avec toutes les valeurs contenues dans les cookies envoyés par le fureteur.
  - Pour accéder au cookie couleur, on écrit $_COOKIE['couleur']
- Il est possible, avec la fonction setcookie(), de spécifier des tableaux de cookies, seulement en donnant un nom de tableau plutôt qu'un nom normal. On accède à ces tableaux de cookies d'une façon spécifique.
  - setcookie(“font[taille]”, $_POST[taille_sel], time() + 3600);
  - Pour accéder à taille, on fait $_COOKIE['font']['type'].

Utilisation des variables sessions

- Comme nous l'avons vu plus tôt, les sessions sont identifiées par un identifiant de session, qui sera généralement contenu dans le cookie du côté client.
- Cet identifiant est généré lors de l'appel de start_session(), qui doit être effectué *avant tout HTML* sur toutes les pages qui utiliseront la session. Cette fonction crée la session si elle n'existe pas, ou trouve la session existante si elle existe.
- Ensuite, il est possible de déclarer et de remplir des variables sessions en utilisant la superglobale $_SESSION[].
  - $_SESSION[“UserID”] = “13541”;
- Cette information pourra donc être utilisée dans d'autres requêtes sur d'autres pages qui partagent cette session. Avant de mettre des valeurs dans une variable session, vous pouvez vérifier si elle a déjà été initialisé avec isset(), comme avec les autres superglobales.
- Si vous voulez détruire une variable session, utiliser unset().
  - unset($_SESSION[“UserID”]);
- Si vous voulez terminer une session (logout), utilisez la fonction session_destroy().