Les sessions et les cookies

Dr N. BAME

Introduction

- Les arrays $_GET et $_POST sont des variables un peu particulières :
  - ✓ leur nom est écrit en majuscules et commence par un underscore,
  - ✓ ces variables sont générées automatiquement par PHP.

➔ Ce sont ce qu'on appelle des variables ***superglobales***

- Il existe d'autres types de variables superglobales. Parmi elles,
  - ✓ certaines permettent de **stocker des informations pendant la durée d'une visite**, c'est le principe des ***sessions***,
  - ✓ mais aussi de stocker des informations **sur l'ordinateur du visiteur** pendant plusieurs mois, c'est le principe des ***cookies***.

Les variables superglobales

sont des variables un peu particulières pour trois raisons :

1. elles sont écrites en *majuscules* et commencent toutes, par un *underscore* (_),
2. sont des *array* car elles contiennent généralement de nombreuses informations,
3. sont *automatiquement créées* par PHP à chaque fois qu'une page est chargée.
    - Elles existent donc sur toutes les pages du site et sont accessibles partout

- On peut utiliser la fonction ***print_r*** pour afficher le contenu d'une variable superglobale

Quelques superglobales (1)

- ***$_SERVER*** : valeurs renvoyées par le serveur.
  - ✓Elles sont nombreuses et quelques-unes d'entre elles peuvent nous être d'une grande utilité.
    - $_SERVER['REMOTE_ADDR'] donne l'adresse IP du client qui a demandé la page.

- ***$_ENV*** : variables d'environnement toujours données par le serveur.
  - ✓C'est le plus souvent sous des serveurs Linux que l'on retrouve des informations dans cette superglobale.

- ***$_FILES*** : contient la liste des fichiers qui ont été envoyés via le formulaire précédent.

Quelques superglobales (2)

- ***$_GET*** : contient les données envoyées en paramètres dans l'URL.

- ***$_POST*** : contient les informations qui viennent d'être envoyées par un formulaire.

- ***$_SESSION*** : variables de session.
  - ✓ Ce sont des variables qui restent stockées sur le serveur le temps de la présence d'un visiteur.

- ***$_COOKIE*** **:** valeurs des cookies enregistrés sur l'ordinateur du visiteur.
  - ✓ permet de stocker des informations sur l'ordinateur du visiteur pendant plusieurs mois.

Les sessions

- un moyen de conserver des **variables** sur **toutes** les pages du site.
  - ✓ Contrairement à $_GET et $_POST qui transmettent d'une page à une autre seulement

- Les sessions sont très utiles dans la mesure où elles sont les seules à permettre un suivi fiable du visiteur dans un site.
  - Les sessions permettent d'enregistrer des variables propres à un utilisateur.

- Une session est caractérisée par :
  - ✓ Un identificateur de session
  - ✓ Des variables associées à la session

Identificateur de session

- Une session est créée avec l'appel : session_start().

- La fonction session_start() essaye de trouver un identificateur de session existant,
  - ✓ sinon une nouvelle session est créée et un identificateur unique est assigné par PHP.

- Toutes les pages utilisant les sessions doivent donc appeler cette fonction pour indiquer au moteur PHP de récupérer les informations relatives à la session.

- La session, à proprement parlé, est stockée côté serveur : seul l'identificateur de la session se trouve côté client.

- La fonction session_start() essaye donc de trouver un identificateur de session dans les cookies.

Les variables de session

- Une variable de session est une variable régulière globale qui, quand elle est enregistrée, garde sa valeur sur toutes les pages qui utilisent les sessions

- Concrètement, une fois qu'une variable de session est créée, il suffit de ré-ouvrir la session pour y avoir directement accès, sans même avoir besoin de la déclarer ou de l'importer.

- Les variables de sessions sont enregistrées sur le serveur.
  - ✓ Cependant, on doit avoir l'identificateur de session pour retrouver les variables de l'utilisateur associées à la session.

Fonctionnement des sessions

Trois étapes

1. Un visiteur ouvre une page du site, on demande à créer une session pour lui. PHP génère alors un numéro unique pour la session.
   - ✓ Ce numéro sert d'identifiant et est appelé «ID de session» (ou ***PHPSESSID***).
   - ✓ PHP transmet automatiquement cet ID de page en page en utilisant généralement un cookie.

2. Une fois la session générée, on peut créer une infinité de variables de session en fonction des besoins.
   - ✓ Par exemple, on peut créer une variable **$_SESSION['nom']** qui contient le nom du visiteur.
   - ✓ Le serveur conserve ces variables même lorsque la page PHP a fini d'être générée.

3. Lorsque le visiteur se déconnecte du site, la session est fermée et PHP «oublie» alors toutes les variables de session qu'on a créées.

Deux fonctions sur les sessions

- *session_start():* démarre le système de sessions.
  - Si le visiteur vient d'arriver sur le site, alors un numéro de session est généré pour lui.
  - Cette fonction doit être appelée au tout début de chacune des pages où on a besoin des variables de session.

- *session_destroy():* ferme la session du visiteur.
  - Cette fonction est automatiquement appelée lorsque le visiteur ne charge plus de page du site pendant plusieurs minutes.

- **Remarque** :
  - il faut appeler session_start() sur *chacune des pages* AVANT d'écrire le moindre code HTML.
    - Si on oublie de lancer session_start(), on ne pourra pas accéder aux variables superglobales $_SESSION.

Utilité des sessions en pratique

- Se souvenir de l'identifiant du visiteur, déjà authentifié, sur toutes les pages du site

- Restreindre certaines pages de notre site à certains visiteurs uniquement.

- gérer un « panier » : on retient les produits que commande le client quelle que soit la page où il est. Lorsqu'il valide sa commande, on récupère ces informations et… on le fait payer

Exercices

- Afficher le nom et prénom de l'utilisateur après authentification sur les toutes les pages
- Restriction de l'accès aux pages d'ajout et de modification aux visiteurs non identifiés

Les cookies

- Un cookie, c'est un petit fichier que l'on enregistre sur l'ordinateur du visiteur.

- Ce fichier contient du texte et permet de « retenir » des informations sur le visiteur.
  - Par exemple, on inscrit dans un cookie le pseudo du visiteur, comme ça la prochaine fois qu'il viendra sur le site, on pourra lire son pseudo en allant regarder ce que son cookie contient.
  - Les cookies ne sont pas des virus, juste de petits fichiers texte qui permettent de retenir des informations.

- Chaque cookie stocke généralement une information à la fois.
  - Si on veut stocker le pseudonyme du visiteur et sa date de naissance, il est donc recommandé de créer deux cookies.
- Pour voir les cookies du navigateur :

  Outils / Options / Vie privée / Supprimer des cookies spécifiques

Écrire un cookie

- Comme une variable, un cookie a un nom et une valeur.
  - Par exemple, le cookie login aurait la valeur *mdiallo*.
- Pour écrire un cookie, on utilise la fonction PHP ***setcookie*** avec trois paramètres, dans l'ordre suivant :
  - 1. le **nom** du cookie (ex. : login) ;
  - 2. la **valeur** du cookie (ex. : *mdiallo*) ;
  - 3. la **date d'expiration** du cookie, sous forme de timestamp (ex. : 1090521508).
- Exemple

```
<? php setcookie('login', 'mdiallo ', time() + 365*24*3600); ?>
```

- Créer le cookie avant d'écrire du HTML
  - Comme pour session_start, la fonction setcookie ne marche QUE si elle est appelée avant tout code HTML.

Lire un cookie

- Avant de commencer à travailler sur une page, PHP lit les cookies du client pour récupérer toutes les informations qu'ils contiennent.

- Ces informations sont placées dans la superglobale **$_COOKIE**, sous forme d'array.
  - De ce fait, si on veut ressortir le login du visiteur qu'on avait inscrit dans un cookie, il suffit d'écrire :

    $_COOKIE['login'].

- Exemple

```
<p>
    Bonjour ! <br />
    Ton nom d'utilisateur est : <?php echo $_COOKIE['login']; ?>
</p>
```

- Il faut faire un isset pour vérifier si le cookie existe ou non.

Modifier un cookie existant

- Il faut refaire appel à setcookie en gardant le même nom de cookie, ce qui « écrasera » l'ancien.

Exercice

- Créer un cookie qui enregistre le login de l'utilisateur une fois que celui-ci s'est authentifié correctement.
- Ce cookie devra pré-remplir le champ login du formulaire d'authentification lors de la prochaine connexion.