Gestion de session et authentification

Les sessions

Source : https://openclassrooms.com/courses/concevez-votre-site-web-avec-php-et-mysql/variables-superglobales-sessions-et-cookies

Les sessions constituent un moyen de conserver des variables sur toutes les pages de votre site. Jusqu'ici, nous étions parvenus à passer des variables de page en page via la méthode `GET` (en modifiant l'URL : `page.php?variable=valeur`) et via la méthode `POST` (à l'aide d'un formulaire). Mais imaginez maintenant que vous souhaitez transmettre des variables sur toutes les pages de votre site pendant la durée de la présence d'un visiteur. Ce ne serait pas facile avec `GET` et `POST` car ils sont plutôt faits pour transmettre les informations une seule fois, d'une page à une autre. On sait ainsi envoyer d'une page à une autre le nom et le prénom du visiteur, mais dès qu'on charge une autre page ces informations sont « oubliées ». C'est pour cela qu'on a inventé les sessions.

Fonctionnement des sessions

Comment sont gérées les sessions en PHP ? Voici les trois étapes à connaître.

1. Un visiteur arrive sur votre site. On demande à créer une session pour lui. PHP génère alors un numéro unique. Ce numéro est souvent très gros et écrit en hexadécimal, par exemple : a02bbffc6198e6e0cc2715047bc3766f.
   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 pour nos besoins. Par exemple, on peut créer une variable `$_SESSION['nom']` qui contient le nom du visiteur, `$_SESSION['prenom']` qui contient le prénom, etc. Le serveur conserve ces variables même lorsque la page PHP a fini d'être générée. Cela veut dire que, quelle que soit la page de votre site, vous pourrez récupérer par exemple le nom et le prénom du visiteur via la superglobale `$_SESSION` !
3. Lorsque le visiteur se déconnecte de votre site, la session est fermée et PHP « oublie » alors toutes les variables de session que vous avez créées. Il est en fait difficile de savoir précisément quand un visiteur quitte votre site. En effet, lorsqu'il ferme son navigateur ou va sur un autre site, le vôtre n'en est pas informé. Soit le visiteur clique sur un bouton « Déconnexion » (que vous aurez créé) avant de s'en aller, soit on attend quelques minutes d'inactivité pour le déconnecter automatiquement : on parle alors de **timeout**. Le plus souvent, le visiteur est déconnecté par un timeout.

Tout ceci peut vous sembler un peu compliqué, mais c'est en fait très simple à utiliser. Vous devez connaître deux fonctions :

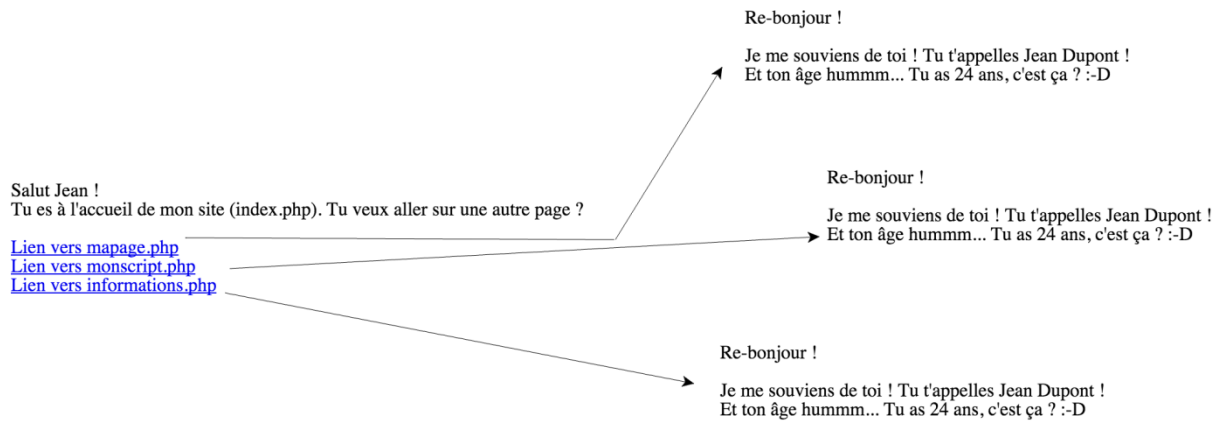
- `session_start()` : démarre le système de sessions. Si le visiteur vient d'arriver sur le site, alors un numéro de session est généré pour lui. Vous devez appeler cette fonction au tout début de chacune des pages où vous avez besoin des variables de session.
- `session_destroy()` : ferme la session du visiteur. Cette fonction est automatiquement appelée lorsque le visiteur ne charge plus de page de votre site pendant plusieurs minutes (c'est le timeout), mais vous pouvez aussi créer une page « Déconnexion » si le visiteur souhaite se déconnecter manuellement.

Il y a un petit piège : il faut appeler `session_start()` sur chacune de vos pages AVANT d'écrire le moindre code HTML (avant même la balise `<!DOCTYPE>`). Si vous oubliez de lancer `session_start()`, vous ne pourrez pas accéder aux variables superglobales `$_SESSION`.

Exemple d'utilisation des sessions

Je vous propose d'étudier un exemple concret pour que vous voyiez à quel point c'est simple à utiliser :

```php
<?php
// On démarre la session AVANT d'écrire du code HTML
session_start();

// On s'amuse à créer quelques variables de session dans $_SESSION
$_SESSION['prenom'] = 'Jean';
$_SESSION['nom'] = 'Dupont';
$_SESSION['age'] = 24;
?>

<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Titre de ma page</title>
    </head>
    <body>

    <p>
        Salut <?php echo $_SESSION['prenom']; ?> !<br />
        Tu es à l'accueil de mon site (index.php). Tu veux aller sur une autre page ?
    </p>

    <p>
        <a href="mapage.php">Lien vers mapage.php</a><br />
        <a href="monscript.php">Lien vers monscript.php</a><br />
        <a href="informations.php">Lien vers informations.php</a>
    </p>

    </body>
</html>
```



Sessions

Ne vous y trompez pas : on peut créer les variables de session n'importe où dans le code (pas seulement au début comme je l'ai fait ici). La seule chose qui importe, c'est que le `session_start()` soit fait au tout début de la page.

Comme vous le voyez, j'ai créé trois variables de session qui contiennent ici le nom, le prénom et l'âge du visiteur.

J'ai aussi fait des liens vers d'autres pages de mon site. Notez quelque chose de très important : ces liens sont tout simples et ne transmettent aucune information. Je ne m'occupe de rien : ni de transmettre le nom, le prénom ou l'âge du visiteur, ni de transmettre l'ID de session. PHP gère tout pour nous.

Maintenant, sur toutes les pages de mon site (bien entendu, il faudra démarrer le système de session sur toutes les pages avec `session_start()`), je peux utiliser si je le souhaite les variables `$_SESSION['prenom']`, `$_SESSION['nom']` et `$_SESSION['age']` !

Voici par exemple le code source de la page `informations.php` :

```php
<?php
session_start(); // On démarre la session AVANT toute chose
?>

<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Titre de ma page</title>
    </head>
    <body>

    <p>Re-bonjour !</p>

    <p>
        Je me souviens de toi ! Tu t'appelles <?php echo $_SESSION['prenom'] . ' ' .
$_SESSION['nom']; ?> !<br />
        Et ton âge hummm... Tu as <?php echo $_SESSION['age']; ?> ans, c'est ça ? :-D
    </p>

    </body>
</html>
```

Vous voyez ? On a juste fait démarrer la session avec un `session_start()`, puis on a affiché les valeurs des variables de session.
Et là, magie !
Les valeurs des variables ont été conservées, on n'a rien eu à faire !
En résumé, on peut créer des variables de session comme on crée des variables classiques, à condition de les écrire dans l'array `$_SESSION` et d'avoir lancé le système de sessions avec `session_start()`.
Ces variables sont ainsi conservées de page en page pendant toute la durée de la présence de votre visiteur.
Si vous voulez détruire manuellement la session du visiteur, vous pouvez faire un lien « Déconnexion » amenant vers une page qui fait appel à la fonction `session_destroy()`. Néanmoins, sachez que sa session sera automatiquement détruite au bout d'un certain temps d'inactivité.

L'utilité des sessions en pratique

Concrètement, les sessions peuvent servir dans de nombreux cas sur votre site (et pas seulement pour retenir un nom et un prénom !). Voici quelques exemples :

- Imaginez un script qui demande un login et un mot de passe pour qu'un visiteur puisse se « connecter » (s'authentifier). On peut enregistrer ces informations dans des variables de session et se souvenir de l'identifiant du visiteur sur toutes les pages du site !
- Puisqu'on retient son login et que la variable de session n'est créée que s'il a réussi à s'authentifier, on peut l'utiliser pour restreindre certaines pages de notre site à certains visiteurs uniquement. Cela permet de créer toute une zone d'administration sécurisée : si la variable de session login existe, on affiche le contenu, sinon on affiche une erreur. Cela devrait vous rappeler le TP « page protégée par mot de passe », sauf qu'ici on peut se servir des sessions pour protéger automatiquement plusieurs pages.
- On se sert activement des sessions sur les sites de vente en ligne. Cela permet de 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. ;-)

Si votre site est hébergé chez Free.fr, vous devrez créer un dossier appelé « sessions » à la racine de votre FTP pour activer les sessions.

Autre exemple synthétique de l'implémentation de session avec authentification

Source :

http://stackoverflow.com/questions/1243150/php-sessions-to-authenticate-user-on-login-form

```
session_start();

if( isset($_POST['username']) && isset($_POST['password']) )

{

    if( auth($_POST['username'], $_POST['password']) )

    {

        // auth okay, setup session

        $_SESSION['user'] = $_POST['username'];

        // redirect to required page

        header( "Location: index.php" );

    } else {

        // didn't auth go back to loginform

        header( "Location: loginform.html" );

    }

 } else {

    // username and password not given so go back to login

    header( "Location: loginform.html" );

 }
```

and at the top of each "secure" page use this code:

```
session_start();

session_regenerate_id();

if(!isset($_SESSION['user']))      // if there is no valid session

{

    header("Location: loginform.html");

}
```

this keeps a very small amount of code at the top of each page instead of running the full auth at the top of every page. To logout of the session:

```
session_start();

unset($_SESSION['user']);

session_destroy();

header("Location: loginform.html");
```