Concevez votre site web avec PHP et MySQL

30 hours Medium License

Session & Cookies

Les sessions

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 :

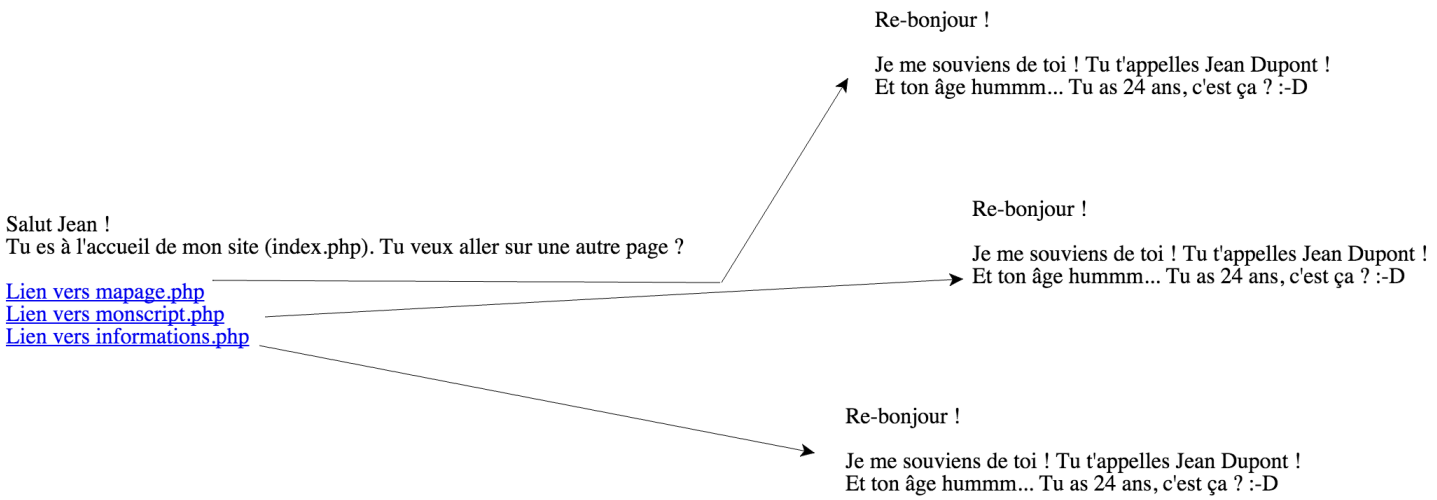
php

```php
<?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
```

```
16      </p>
17      </body>
18  </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.

Les cookies

Travailler avec des cookies revient à peu près à la même chose qu'avec des sessions, à quelques petites différences près que nous allons voir. Voici ce que nous allons faire pour découvrir les cookies :

1. on va voir ce qu'est exactement un cookie (parce que si ça se trouve, il y en a qui croient en ce moment même que je vais parler de recettes de cuisine !) ;
2. ensuite, nous verrons comment **écrire un cookie** : c'est facile à faire, si on respecte quelques règles ;

3. enfin, nous verrons comment **récupérer le contenu d'un cookie** : ce sera le plus simple.

Qu'est-ce qu'un cookie ?

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, vous inscrivez dans un cookie le pseudo du visiteur, comme ça la prochaine fois qu'il viendra sur votre site, vous pourrez lire son pseudo en allant regarder ce que son cookie contient.

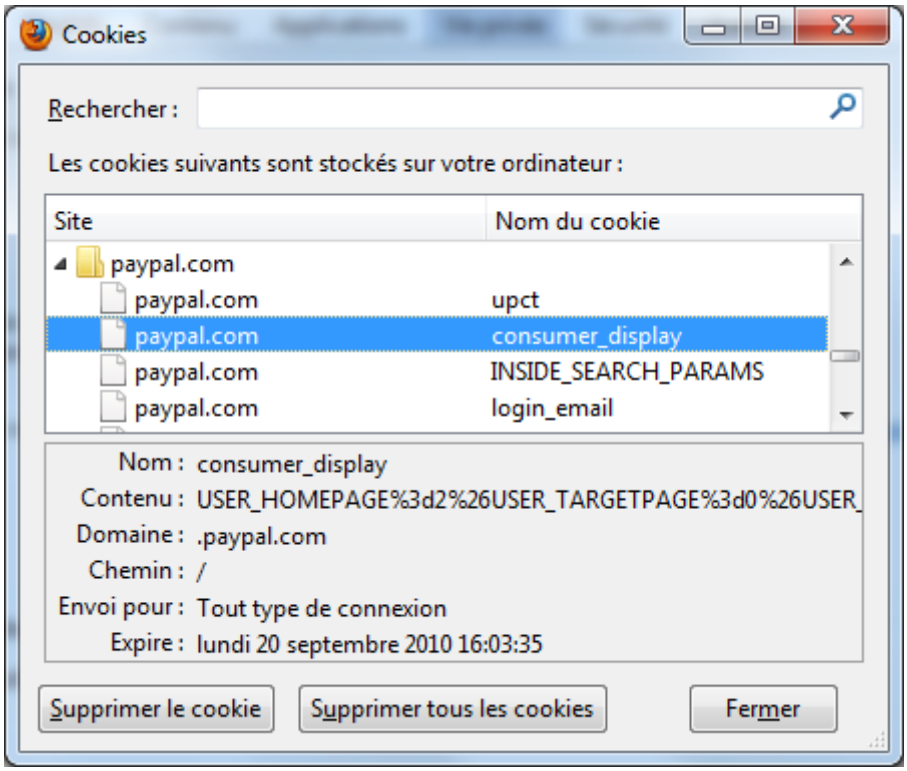
Parfois les cookies ont une mauvaise image. On fait souvent l'erreur de penser que les cookies sont « dangereux ». Non, ce ne sont pas des virus, juste de petits fichiers texte qui permettent de retenir des informations. Au pire, un site marchand peut retenir que vous aimez les appareils photos numériques et vous afficher uniquement des pubs pour des appareils photos, mais c'est tout, ces petites bêtes sont inoffensives pour votre ordinateur.

Chaque cookie stocke généralement une information à la fois. Si vous voulez stocker le pseudonyme du visiteur et sa date de naissance, il est donc recommandé de créer deux cookies.

Où sont stockés les cookies sur mon disque dur ?

Cela dépend de votre navigateur web. Généralement on ne touche pas directement à ces fichiers, mais on peut afficher à l'intérieur du navigateur la liste des cookies qui sont stockés. On peut choisir de les supprimer à tout moment.

Si vous avez Mozilla Firefox, vous pouvez aller dans le menu `Outils / Options / Vie privée` et cliquer sur `Supprimer des cookies spécifiques` . Vous obtenez la liste et la valeur de tous les cookies stockés, comme sur la figure suivante.



Cookies sous Firefox

Les cookies sont classés par site web. Chaque site web peut écrire, comme vous le voyez, plusieurs cookies. Chacun d'eux a un nom et une valeur (que vous pouvez voir à la ligne `Contenu` sur la figure suivante). Vous noterez que comme tout cookie qui se respecte, chacun a une date d'expiration. Après cette date, ils ne sont plus bons à manger ils sont automatiquement supprimés par le navigateur.

Les cookies sont donc des informations temporaires que l'on stocke sur l'ordinateur des visiteurs. La taille est limitée à quelques kilo-octets : vous ne pouvez pas stocker beaucoup d'informations à la fois, mais c'est en général suffisant.

Écrire un cookie

Comme une variable, un cookie a un nom et une valeur. Par exemple, le cookie pseudo aurait chez moi la valeur `M@teo21` .

Pour écrire un cookie, on utilise la fonction PHP `setcookie` (qui signifie « Placer un cookie » en anglais).

On lui donne en général trois paramètres, dans l'ordre suivant :

1. le nom du cookie (ex. : `pseudo`) ;
2. la valeur du cookie (ex. : `M@teo21`) ;
3. la date d'expiration du cookie, sous forme de timestamp (ex. : `1090521508`).

Le paramètre correspondant à la date d'expiration du cookie mérite quelques explications. Il s'agit d'un timestamp, c'est-à-dire du nombre de secondes écoulées depuis le 1er janvier 1970. Le timestamp est une valeur qui augmente de 1 toutes les secondes. Pour obtenir le timestamp actuel, on fait appel à la fonction `time()` . Pour définir une date d'expiration du cookie, il faut ajouter au « moment actuel » le nombre de secondes au bout duquel il doit expirer.

Si vous voulez supprimer le cookie dans un an, il vous faudra donc écrire : `time() + 365*24*3600` . Cela veut dire : timestamp actuel $+$ nombre de secondes dans une année. Cela aura pour effet de supprimer votre cookie dans exactement un an.

Voici donc comment on peut créer un cookie :

php

```
1 <?php setcookie('pseudo', 'M@teo21', time() + 365*24*3600); ?>
```

Sécuriser son cookie avec le mode httpOnly

Je recommande toutefois d'activer l'option `httpOnly` sur le cookie. Sans rentrer dans les détails, cela rendra votre cookie inaccessible en JavaScript sur tous les navigateurs qui supportent cette option (c'est le cas de tous les navigateurs récents.). Cette option permet de réduire drastiquement les risques de faille XSS sur votre site, au cas où vous auriez oublié d'utiliser `htmlspecialchars` à un moment.

Je vous **recommande** donc de créer votre cookie plutôt comme ceci :

php

```
1 <?php setcookie('pseudo', 'M@teo21', time() + 365*24*3600, null, null, false, true); ?>
```

Le dernier paramètre `true` permet d'activer le mode `httpOnly` sur le cookie, et donc de le rendre en quelque sorte plus sécurisé. Ça ne coûte rien et vous diminuez le risque qu'un jour l'un de vos visiteurs puisse se faire voler le contenu d'un cookie à cause d'une faille XSS.

Les paramètres du milieu sont des paramètres que nous n'utilisons pas, je leur ai donc envoyé `null` .

Créer le cookie avant d'écrire du HTML

Il y a un petit problème avec `setcookie` … Comme pour `session_start` , cette fonction ne marche QUE si vous l'appelez avant tout code HTML (donc avant la balise `<!DOCTYPE>`).

Ne placez donc JAMAIS le moindre code HTML avant d'utiliser `setcookie` . La plupart des gens qui ont des problèmes avec `setcookie` ont fait cette erreur, donc souvenez-vous en !

Voyons maintenant comment je ferais pour écrire deux cookies, un qui retient mon pseudo pendant un an, et un autre qui retient le nom de mon pays :

php

```php
<?php
setcookie('pseudo', 'M@teo21', time() + 365*24*3600, null, null, false, true); // On écrit un cookie
setcookie('pays', 'France', time() + 365*24*3600, null, null, false, true); // On écrit un autre cookie...

// Et SEULEMENT MAINTENANT, on peut commencer à écrire du code html
?>

<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Ma super page PHP</title>
    </head>
    <body>

    etc.
```

Et voilà, les cookies sont écrits ! Comme vous le voyez, pour écrire deux cookies il faut appeler deux fois `setcookie` .

Afficher un cookie

C'est la partie la plus simple. 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, comme d'habitude.

De ce fait, si je veux ressortir le pseudo du visiteur que j'avais inscrit dans un cookie, il suffit d'écrire : `$_COOKIE['pseudo']` .

Ce qui nous donne un code PHP tout bête pour afficher de nouveau le pseudo du visiteur :

php

```php
<p>
    Hé ! Je me souviens de toi !<br />
    Tu t'appelles <?php echo $_COOKIE['pseudo']; ?> et tu viens de <?php echo $_COOKIE['pays']; ?> c'est bien ça ?
</p>
```

Comme vous le voyez encore une fois, le gros avantage c'est que les superglobales sont accessibles partout. Vous avez besoin de savoir ce que contient le cookie `pseudo` ? Affichez donc le contenu de la superglobale `$_COOKIE['pseudo']` !

À noter que si le cookie n'existe pas, la variable superglobale n'existe pas. Il faut donc faire un `isset` pour vérifier si le cookie existe ou non.

Les cookies viennent du visiteur. Comme toute information qui vient du visiteur, **elle n'est pas sûre**. N'importe quel visiteur peut créer des cookies et envoyer ainsi de fausses informations à votre site. Souvenez-vous en lorsque vous lisez les cookies du visiteur : il peut les avoir modifiés, donc soyez prudents et n'ayez pas une confiance aveugle en leur contenu !

Modifier un cookie existant

Vous vous demandez peut-être comment modifier un cookie déjà existant ? Là encore, c'est très simple : il faut refaire appel à `setcookie` en gardant le même nom de cookie, ce qui « écrasera » l'ancien.

Par exemple, si j'habite maintenant en Chine, je ferai :

```php
<?php setcookie('pays', 'Chine', time() + 365*24*3600, null, null, false, true); ?>
```

Notez qu'alors le temps d'expiration du cookie est remis à zéro pour un an.

En résumé

- Les variables superglobales sont des variables automatiquement créées par PHP. Elles se présentent sous la forme d'arrays contenant différents types d'informations.
- Dans les chapitres précédents, nous avons découvert deux superglobales essentielles : `$_GET` (qui contient les données issues de l'URL) et `$_POST` (qui contient les données issues d'un formulaire).
- La superglobale `$_SESSION` permet de stocker des informations qui seront automatiquement transmises de page en page pendant toute la durée de visite d'un internaute sur votre site. Il faut au préalable activer les sessions en appelant la fonction `session_start()`.
- La superglobale `$_COOKIE` représente le contenu de tous les cookies stockés par votre site sur l'ordinateur du visiteur. Les cookies sont de petits fichiers que l'on peut écrire sur la machine du visiteur pour retenir par exemple son nom. On crée un cookie avec la fonction `setcookie()`.

Que pensez-vous de ce cours ?

Variables superglobales

Lire et écrire dans un fichier

The author

Mathieu Nebra

Entrepreneur à plein temps, auteur à plein temps et co-fondateur d'OpenClassrooms :o)

Check out this course via

eBook


Livre papier

PDF

Vidéo