Sessions et cookies pour l'authentification serveur

Technologies web

Chris Chevalier - chevalier@chris-freelance.com

L'utilisation de cookies avec PHP

Sessions et cookies pour l'authentification serveur

Chris Chevalier - chevalier@chris-freelance.com

Les limites du protocole HTTP

- **HTTP** est très utile pour échanger des informations entre **client** et **serveur**
- Néanmoins, ce **protocole** a ses limites et ne permet pas de conserver les **requêtes** passées au **serveur** - Il exécute chacune des **requêtes** de manière indépendante
- Il nous est donc impossible de conserver des informations en se basant uniquement sur ce **protocole**
- Or, dans plusieurs cas de figure, il peut être très intéressant de conserver les **informations** et **actions** réalisées par le **client**
- Son **identification**, son **statut** d'administrateur ou de simple visiteur, son **parcours** sur le site, les **choix** qu'il a fait concernant un **panier d'achat** ou encore des **réglages**…

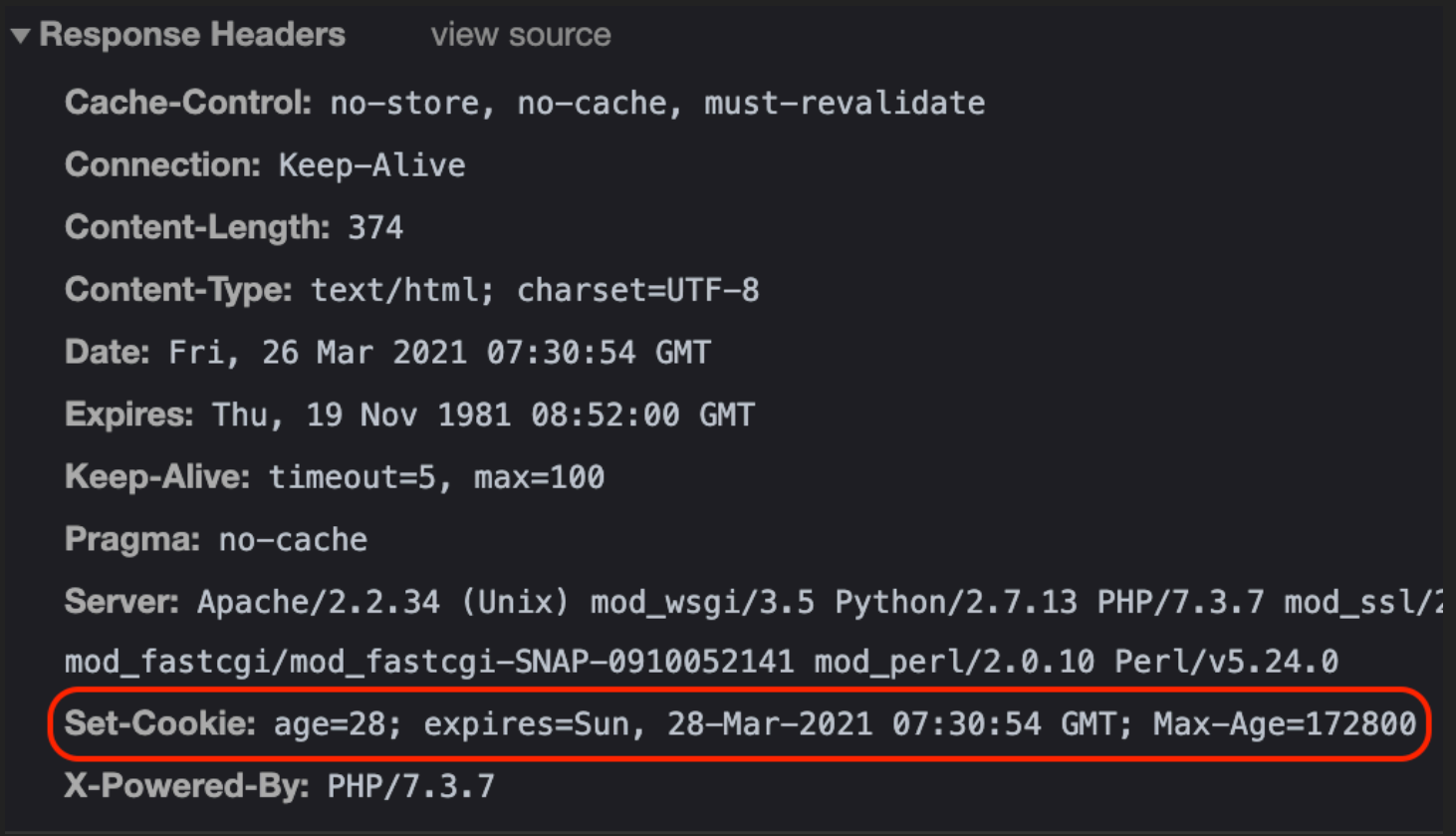
Des solutions

- Il existe plusieurs solutions pour palier ce problème inhérent aux échanges **HTTP**
- Il est possible de passer des informations en **paramètre de l'URL** mais celle-ci est **limitée** en taille et les valeurs passées sont **visibles dans la barre d'adresse** du navigateur
- La solution à privilégier consiste à utiliser des **cookies**

Les cookies

- Les **cookies HTTP** ont été mis en place pour la première fois en **1994** pour gérer un **panier d'achat**
- Ils ont finalement été ajoutés au **protocole HTTP** en **1997**
- Un **cookie** représente un fichier stockant des données chez le **client** par le **serveur**
- Ainsi, ces informations sont transmises dans l'**en-tête HTTP** au **serveur** à chaque **requête**
- Les **cookies** sont, par ailleurs, conservés même si le visiteur quitte le site web concerné

Les cookies


Response Headers view source
Cache-Control: no-store, no-cache, must-revalidate
Connection: Keep-Alive
Content-Length: 374
Content-Type: text/html; charset=UTF-8
Date: Fri, 26 Mar 2021 07:30:54 GMT
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Keep-Alive: timeout=5, max=100
Pragma: no-cache
Server: Apache/2.2.34 (Unix) mod_wsgi/3.5 Python/2.7.13 PHP/7.3.7 mod_ssl/
mod_fastcgi/mod_fastcgi-SNAP-0910052141 mod_perl/2.0.10 Perl/v5.24.0
Set-Cookie: age=28; expires=Sun, 28-Mar-2021 07:30:54 GMT; Max-Age=172800
X-Powered-By: PHP/7.3.7

Le fonctionnement des cookies (1/2)

- Plus concrètement, le **client** demande une **page** au **serveur**

```
GET /index.html HTTP/1.1
Host: www.example.org
```

- Le **serveur** lui envoie la **page** souhaitée et demande au **navigateur** de créer un **cookie** - Ce traitement doit bien entendu être écrit en **PHP**

```
HTTP/1.0 200 OK
Content-type: text/html
Set-Cookie: name=value
Set-Cookie: name2=value2; Expires=Wed, 09 Jun 2021 10:18:14 GMT

<!DOCTYPE html>
<html lang=fr>
(suite de la page…)
```

- La **navigateur** stocke la ou les informations dans un **cookie** valable jusqu'à la **date** demandée

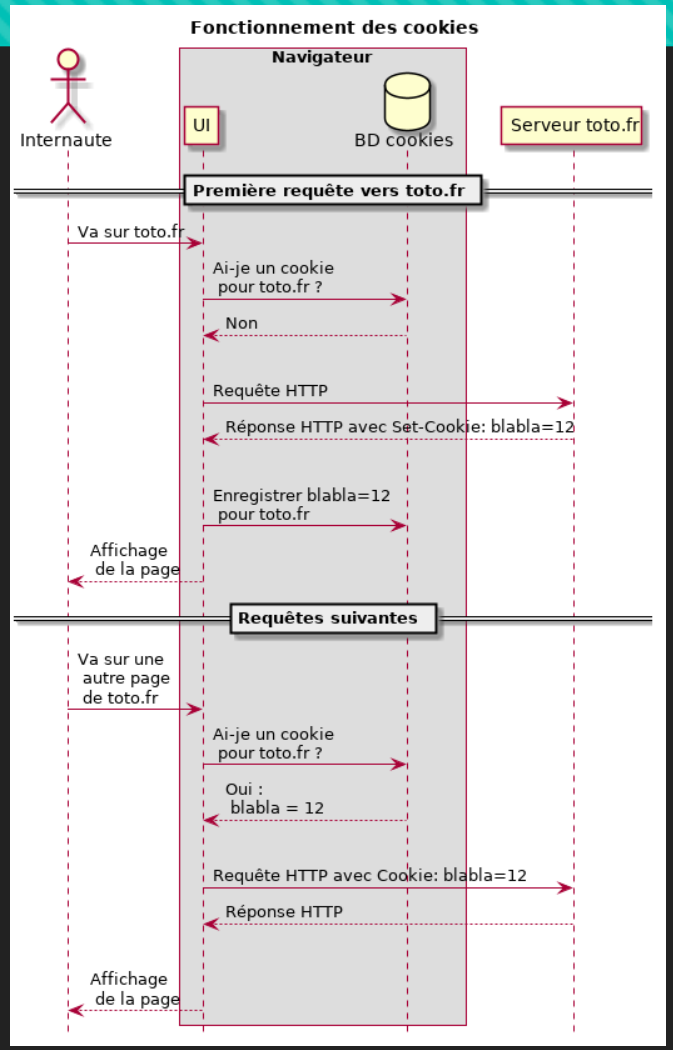
Le fonctionnement des cookies (2/2)

- Ainsi, chaque nouvelle **requête** du **client** vers ce même **serveur** contiendra le contenu du **cookie** stocké dans le **navigateur**

```
GET /toto.html HTTP/1.1
Host: www.example.org
Cookie: name=value; name2=value2
```

- Le **serveur** peut néanmoins choisir de modifier ce contenu avec l'**en-tête de réponse HTTP** `Set-Cookie`

Diagramme de séquence du fonctionnement des cookies



Les cookies et PHP

- **PHP** possède plusieurs **fonctions** natives permettant la manipulation de **cookies**
- Elles s'occupent, en réalité, de modifier une **requête HTTP** pour y ajouter l'**en-tête de réponse** `Set-Cookie`
- Elles permettent également de lire les informations contenues dans le champ `Cookie` d'une **requête**
- Ainsi, pour créer un nouveau **cookie**, on utilisera la **fonction** `setcookie()`

```
setcookie("prenom", "Chris", time() + (86400 * 7));
```

Explications

- Avec cette ligne de code **PHP**, nous avons initialisé un **cookie** valable **7 jours** (**7 x 86400 secondes**, soit **7 x 24h**) et contenant l'information `prenom=Chris`

- En **PHP**, les **cookies** envoyés par le **client**, sont accessibles depuis la **variable superglobale** `$_COOKIE` - Il s'agit d'un **tableau associatif**

```php
if(key_exists("prenom", $_COOKIE)) {
    echo "Bienvenue {$_COOKIE['prenom']} !";
}
```

- Ici, on vérifie que la **clé** `prenom` existe dans le **tableau** `$_COOKIE` avant d'afficher un message de bienvenue contenant le prénom stocké dans le **cookie**

- La vérification de l'existence du **cookie** est essentielle étant donné que le **client** et le **navigateur** peuvent les supprimer

Les limites des cookies

- Les **cookies** s'avèrent, effectivement, plus pratique à utiliser que les **paramètres de l'URL** - Malgré tout, ils présentent certaines **limitations**
- Le **client** y a accès et peut les modifier
- C'est pour cette raison que l'on se contentera bien souvent d'**identifier le client** (en l'associant à un numéro par exemple) - Les autres informations devront donc être stockés sur le **serveur**
- Il peut être lourd et peu pertinent de stocker ce genre de petites informations en **BDD**
- **PHP** permet justement de stocker ces informations côté **serveur** sans passer par une **BDD** - Pour cela, on utilisera les **sessions**

L'utilisation de sessions avec PHP

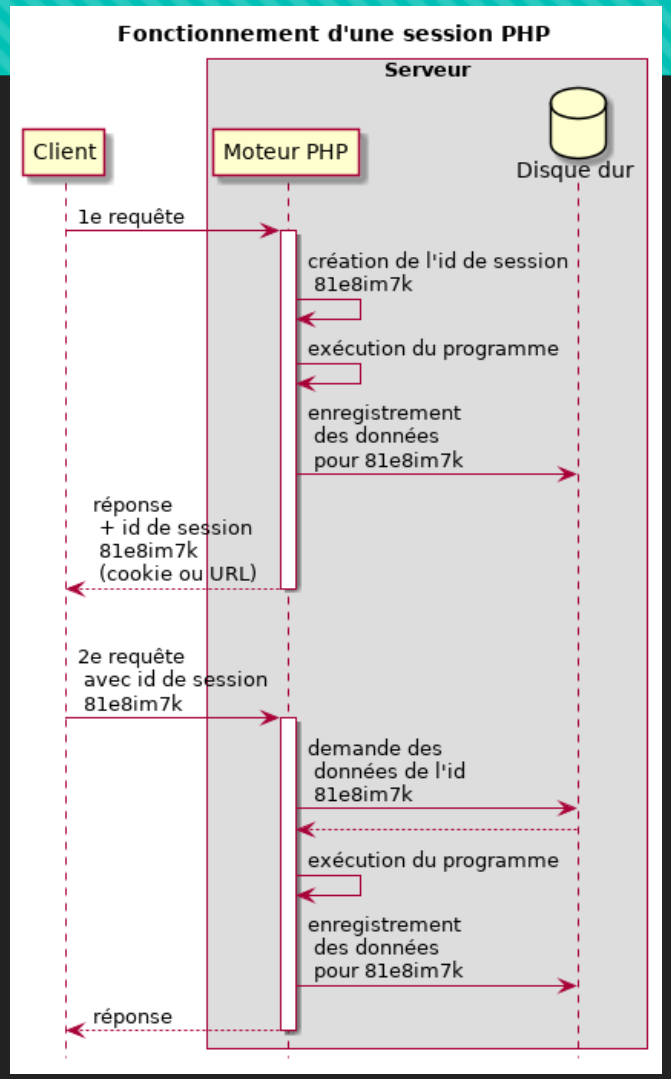
Sessions et cookies pour l'authentification serveur

Chris Chevalier - chevalier@chris-freelance.com

Les sessions

- Les **sessions** permettent de conserver des **variables** d'une **page** à l'autre de façon transparente
- Lors de l'initialisation d'une **session**, le **serveur** génère un **identifiant** unique transmis au **client** via un **cookie** (nommé `PHPSESSID` par défaut)
- Cet **identifiant** est envoyé au **serveur** à chaque **requête** du **client** - Ainsi, le **serveur** utilise les données liées à ce **client**
- Comme les **cookies**, les **sessions** ont une **durée de vie** configurable mais **limitée** - Les données sont supprimées du **serveur** en cas d'inactivité

Diagramme de séquence du fonctionnement des sessions



Une session sans cookie

- Comme nous l'avons vu précédemment, l'**identifiant de session** est normalement stocké dans un **cookie** supprimé à la fermeture du **navigateur**
- Or, si le **client** refuse les **cookies**, la **session** ne fonctionne plus…
- Le cas échéant, le **serveur** se basera sur la **méthode** des « **URLs longues** » - Toutes les **URLs** et tous les **liens locaux** seront réécrits automatiquement afin d'y ajouter un **paramètre** contenant l'**identifiant de la session**
- Il s'agit néanmoins d'une pratique **moins sécurisée** étant donnée que notre site sera vulnérable à la **faille de sécurité** de **fixation de session**
- Pour éviter cela, on peut forcer l'utilisation de **cookies** avec l'**option PHP** `session.use_only_cookies`

Les sessions et PHP (1/2)

- Pour **démarrer une session** en **PHP**, il nous suffira d'utiliser la **fonction** `session_start()` sur chaque page
- Si aucune session n'existe, l'**identifiant** sera généré et un fichier de données est stocké sur le **serveur**
- Si la **session** existe déjà, les **variables** enregistrées en **session** sont chargées dans le **tableau associatif** `$_SESSION`
- Une fois le script **PHP** terminé, le contenu du **tableau** `$_SESSION` est sauvegardé sur le **serveur**

Les sessions et PHP (2/2)

```php
<?php
    session_start(); // On démarre la session
    // Si les valeurs à faire apparaître sont dans la session, on les affiche
    if(key_exists("prenom", $_SESSION) && key_exists("nom", $_SESSION) && key_exists("date", $_SESSION)) {
        // On affiche les différentes valeurs stockées dans la session
        echo "<p>Bienvenue {$_SESSION['prenom']} {$_SESSION['nom']} !</p>";
        echo "<p>Dernière connexion : {$_SESSION['date']}</p>";
        // On met à jour la valeur de la date avec la date et l'heure actuelles
        $_SESSION["date"] = date("Y-m-d H:i:s");
    } else { // Si une des valeurs n'apparaît pas dans la session, on affiche un message différent
        echo "Vous n'êtes pas connecté...";
    }
?>
```

Les sessions utilisateur

- Les **sessions PHP** sont utilisées pour implémenter des **sessions utilisateur**
- Le **client** s'authentifie et le **serveur** stocke ses données dans la **session** (`$_SESSION`)
- Il est néanmoins préférable d'utiliser un **système d'authentification** se référant à une **BDD** et réserver les **sessions PHP** pour d'autres traitements (gérer un **panier d'achat** par exemple)

Quelques notions sur les sessions

- Les **variables** stockées dans la **session** peuvent prendre n'importe quel **type**
- À la fin du script **PHP**, elles sont enregistrées dans un fichier côté **serveur** au format texte - On dit qu'elles sont **sérialisées**
- Elles sont **désérialisées** au chargement des données de **session** (`session_start()`)

Compléments

- Il est possible de supprimer une **variable de session** avec la **fonction** PHP `unset()`
- La **fonction** `session_destroy()` permet de supprimer toutes les **variables de session** du **serveur**
- Il est possible de **sérialiser** et **désérialiser** des **variables de sessions** sous forme de **chaine de caractères** pour les stocker dans une **BDD** par exemple => `session_encode()` et `session_decode()`
- Par défaut le **cookie** stockant l'**id** d'une **session** est nommé `PHPSESSID` - Il est recommandé de le renommer pour une application réelle
- La **fonction** `session_name()` prend un nouveau nom d'**id** en **paramètre** et doit être appelée avant la **fonction** `session_start()`

TP - Session utilisateur

Sessions et cookies pour l'authentification serveur

TP - Session utilisateur - Formulaire HTML

- Dans un fichier `session.php`, créez un **formulaire HTML** avec plusieurs **champs de saisie** :
  - Un champ de type `text` permettant de saisir le **prénom** de l'utilisateur
  - Un champ de type `text` permettant de saisir le **nom** de l'utilisateur
  - Un champ de type `number` permettant de saisir **l'âge** de l'utilisateur

- Donnez une valeur à **l'attribut** `name` de chaque **champ** afin de pouvoir récupérer la donnée saisie simplement une fois le **formulaire** soumis

- La soumission du formulaire devra être faite avec une **méthode** `POST`

TP - Session utilisateur - La connexion

- Une fois le **formulaire HTML** soumis, stockez l'âge saisi dans un **cookie** valable pendant **48 heures** (Cf. diapo 10)

- Si cette valeur est présente dans le **cookie**, affichez l'âge avec un message du type `"Vous avez 28 ans."` (Cf. diapo 11)

- Stockez maintenant les **nom** et **prénom** saisis dans la **session** (Cf. diapo 18)

- Si ces valeurs sont présentes dans la **session**, elles s'affichent dans un **message de bienvenue** du type `"Bienvenue Chris Chevalier !"` - Sinon, un message vous invite à vous connecter via le **formulaire HTML**

TP - Session utilisateur - La déconnexion

- En étant **connecté**, un **bouton** permettant la **déconnection** doit apparaître
- Pour permettre la **déconnexion** de l'utilisateur à la **session**, associez un **lien** au bouton de déconnexion pointant vers un fichier `disconnect.php` situé dans le même répertoire que `session.php`
- Ce fichier ne comporte pas de code HTML mais uniquement un court **script PHP** permettant de **détruire la session** avant de rediriger l'utilisateur vers le fichier `session.php` (Cf. diapo 21)
- Pour rediriger un utilisateur vers un **fichier**, vous pouvez utiliser la **fonction PHP** `header()` - Cette dernière prend une **chaine de caractères** en **argument** => `header('Location: monFichier.php');`

```php
header('Location: index.php');
```

TP - Session utilisateur

- Voici une proposition d'affichage final attendu pour valider le TP :

Pour aller plus loin…

- Documentation officielle **PHP** concernant la **fonction** `setcookie()` : https://www.php.net/manual/fr/function.setcookie.php

- Liste des **options PHP** possibles pour les **sessions** : https://www.php.net/session.configuration

Des questions ?