Sessions PHP

Alexandre Niveau

GREYC — Université de Caen

En partie adapté du cours de Jean-Marc Lecarpentier

Rappels sur les cookies

- Les cookies sont des informations enregistrées côté client
- Transmises au serveur à chaque requête
- Permettent de contourner la conception *sans état* de HTTP
- En particulier, de conserver le *contexte* d'une page à l'autre

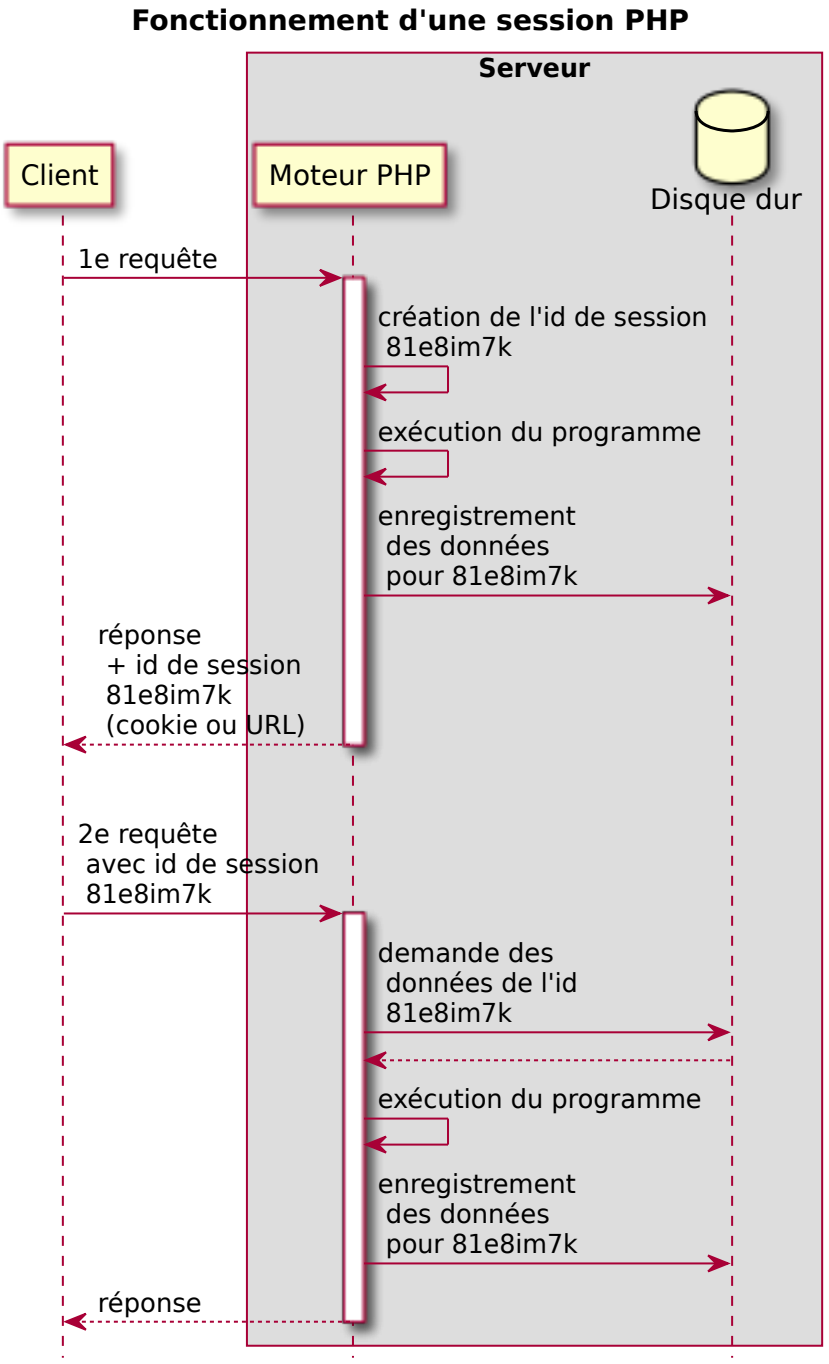
Limites des cookies

- Les cookies sont plus pratiques que l'utilisation de paramètres d'URL ou de champs cachés, mais pas si différents
- En particulier, l'internaute peut les modifier à loisir
- Solution : n'utiliser les cookies que pour *identifier* l'internaute (par exemple en lui associant un numéro), et ne stocker les informations que côté serveur
- Quel stockage côté serveur ? Utiliser une BD est lourd et peu efficace si on veut simplement conserver quelques variables d'une page à l'autre

  ➤ PHP permet de faire ceci simplement et sans passer par une BD grâce aux *sessions*

Principe d'une session PHP

- L'utilisation d'une session revient à pouvoir garder la valeur de *certaines variables* d'une page à l'autre, de manière transparente
- Lors de l'initialisation de la session : génération d'un identifiant unique par le serveur, transmis au client par un cookie
- À chaque nouvelle requête du client, l'identifiant est transmis, et le serveur utilise les données liées à ce client
- Les sessions ont une durée de vie limitée (configurable) : en cas d'inactivité prolongée, les données sont effacées du serveur



Fonctionnement d'une session PHP (lien vers l'image SVG) [img/seq_session.svg] (lien vers l'image PNG) [img/seq_session.png]

Utilisation des sessions PHP

- Il faut explicitement demander l'utilisation d'une session : `session_start()` sur chaque page
- Si aucune session n'existe : génération d'un identifiant, création d'un fichier de données sur le serveur
- Si déjà une session : les variables enregistrées sont chargées en mémoire dans le tableau `$_SESSION`
- À la fin du script, le contenu du tableau `$_SESSION` est sauvegardé sur le serveur

Exemple

```
<?php
```

```php
/* on demande à PHP de remplir $_SESSION
 * avec les données de la session */
session_start();

/* affichage d'une valeur écrite lors d'une précédente requête */
echo "<p>Dernier accès : " .  $_SESSION["date"] . "</p>";

/* modification du tableau, qui persistera lors de futures requêtes */
$_SESSION["date"] = date("Y-m-d H:i:s");
?>
```

Sessions PHP ≠ session utilisateur

- Les sessions PHP sont typiquement utilisées pour implémenter des sessions utilisateur
- On demande à l'internaute de s'authentifier, et une fois que c'est fait, on stocke ses données dans le tableau `$_SESSION`
- Cependant, on peut parfaitement utiliser `$_SESSION` sans implémenter d'authentification, et on peut faire en sorte que la session soit valable pendant des mois, indépendamment de la fermeture de la page (en changeant le *lifetime* du cookie de session via `session_set_cookie_params` [http://fr2.php.net/manual/fr/function.session-set-cookie-params.php])

  ➤ les sessions PHP sont simplement un mécanisme offert par PHP pour simplifier et sécuriser l'utilisation de cookies

Compléments

- Pour supprimer une variable de session : `unset` (comme pour une variable normale)
- Pour effacer toutes les variables de session du serveur : `session_destroy()` (attention, l'effet ne sera visible qu'à la fin du script)
- Il est possible de sérialiser soi-même les variables de session sous forme de chaîne, par exemple pour les stocker dans une BD : `session_encode()`, `session_decode()`

Remarques

- Par défaut, le cookie de session s'appelle `PHPSESSID`
- Il est recommandé de changer ce nom pour une application réelle
- Ceci se fait avec `session_name("monsiteID");`, obligatoirement avant le `session_start();`. Attention : caractères alphanumériques seulement
- Lecture/écriture de cookies : se fait dans l'en-tête HTTP. Donc comme pour la fonction `header`, `setcookie` et `session_start` doivent être appelées **avant tout envoi de contenu** (même une espace ou un saut de ligne !)
- En cas d'erreur ou de warning avec «headers already sent», chercher de ce côté-là

Sessions sans cookies

- L'identifiant de session est normalement stocké dans un cookie qui s'efface à la fermeture du navigateur

➤ Ça ne peut pas marcher si le client refuse les cookies

- Dans ce cas le serveur utilise la technique dite des « URL longues » : toutes les URL des liens locaux sont réécrites automatiquement, pour y ajouter un paramètre contenant l'identifiant de session
- Moins sécurisé, car les attaques de type « *session fixation* » [https://fr.wikipedia.org/wiki/Fixation_de_session] sont plus faciles si l'identifiant est dans l'URL que dans un cookie

  ➤ On peut forcer l'utilisation de cookies avec l'option `session.use_only_cookies`
- Liste des options PHP pour les sessions [http://fr2.php.net/session.configuration]

Sécurité des sessions PHP

- Les paramètres par défaut des sessions PHP ne sont pas toujours les plus recommandés en termes de sécurité
- voir les recommandations du manuel PHP [https://www.php.net/manual/en/session.security.ini.php], les réglages les plus importants étant au début
- d'autres recommandations raisonnables, un peu plus détaillées/justifiées, dans cette réponse sur StackOverflow [https://stackoverflow.com/a/5081453]
- voir aussi un résumé de toutes les attaques possibles sur les sessions [https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html] (très bonne source, mais pas spécifique à PHP)

Références et guides

- Manuel PHP sur les sessions [http://fr2.php.net/manual/fr/intro.session.php]

Lectures complémentaires

- OWASP Session management cheat sheet [https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html]


[http://creativecommons.org/licenses/by-nc-sa/4.0/]