Langage de Script PHP

Partie 3 : Les sessions

Enseignants de l'EMSI RABAT

version éditée par: M. BELATAR

3éme Année - Ingénierie en Informatique et Réseaux

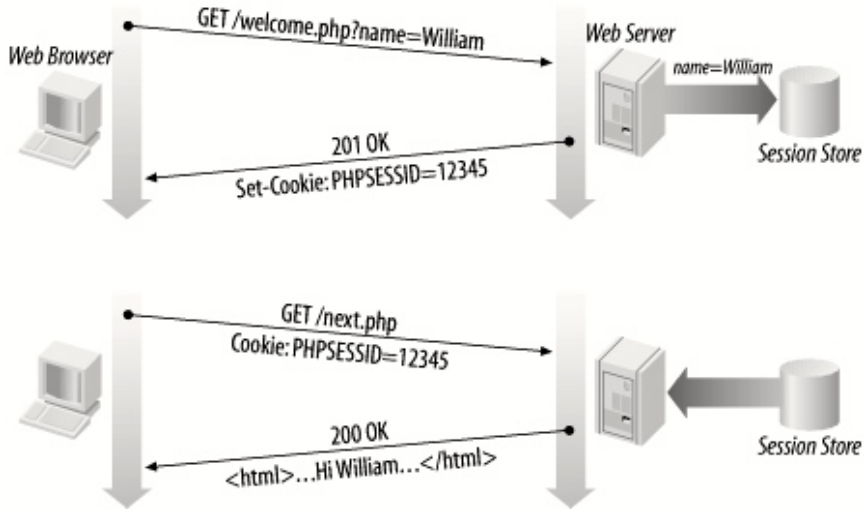
5 décembre 2023

Table of Contents

Table of Contents

1. Un concept permettant de lier une suite de requêtes et réponses HTTP entre un navigateur donné et un serveur.
   - HTTP est un protocole **stateless** et ne permet pas cette liaison par défaut.
2. Implémenté à l'aide des *cookies*.
   - Un *cookie* est une donnée envoyée par le serveur et stockée par le navigateur du client.
   - Une session est représentée par un identifiant unique (généré par le serveur).
   - Cet identifiant est placé dans un cookie stocké chez le client.
   - A chaque requête, le *cookie* de session est envoyé automatiquement par le navigateur au serveur.
   - Le serveur peut stocker des informations associées à l'identifiant de session et ces informations sont persistantes.

Les sessions
Comment les sessions sont-elles établies



- Le navigateur du client envoie une requête initiale de demande de session.
- Le serveur enregistre l'adresse IP et l'identifiant du navigateur et envoie au client un **identifiant de session** sous la forme d'un *cookie*.
- Pour toutes les requêtes suivantes, le navigateur du client envoie automatiquement le cookie correspondant.
- Le serveur utilise l'identifiant de session (reçu via un *cookie*) pour stocker/restitue des données persistantes.

- **Durée de vie** : Les sessions restent actives jusqu'au log out ou jusqu'au moment où l'utilisateur arrête le navigateur. Les *cookies* de session restent stockés sur le disque du client au moins le temps d'activité de la session correspondante.

- **Stockage des données** : Une session stocke des données sur le serveur alors qu'un *cookie* (et les données qu'il contient) est stocké chez le client.

- **Sécurité** : Les sessions sont difficiles à *pirater*. Les cookies sont simplement des fichiers stockés sur le disque du client.

PHP (modèle)

```
session_start();
```

- **session_start()** doit être appelée au début des scripts, avant l'envoi de code HTML.
- Lors du premier appel à **session_start()** le serveur fabrique une nouvelle session et initialise un nouveau tableau associatif global de nom `$_SESSION`.
- Une fois le tableau `$_SESSION` créé, vous pouvez stocker des données dedans, données qui sont enregistrées sur le serveur.
- Lors des appels suivants à **session_start()**, les données persistantes copiées précédemment sont chargées dans le tableau `$_SESSION` et disponibles dans le script. Une fois le script terminé, le contenu du tableau `$_SESSION` est enregistré sur le serveur.

PHP (modèle)

```
$_SESSION["name"] = value;      # stocke une valeur de session
$variable = $_SESSION["name"];  # lit une valeur de session
if (isset($_SESSION["name"]))   # teste si la valeur existe
```

PHP (Exemple)

```
if (isset($_SESSION["name"])) {
  $points = $_SESSION["name"];
    print("You've earned $points points.");
} else {
  $_SESSION["name"] = 0;                    # default
}
```

- `$_SESSION` permet de gérer les données (persistantes) de session.
- **isset** permet des tester si une valeur de *session* existe.

1. Le protocole HTTP étant *stateless*, il est difficile pour le serveur de savoir quand un utilisateur à terminer une session.
2. Il est possible de détruire explicitement une session, en général lors d'un log out, mais beaucoup d'utilisateur n'effectue pas de *log out*.
3. le navigateur du client supprimer les *cookies* de session lorsqu'il est arrêté.
4. Le serveur supprime automatiquement les vieilles sessions après un certain temps :
   - Une session inactive consomme des ressources et peut représenter une faille de sécurité.
   - Il est possible de terminer explicitement une session avec la fonction **session_destroy()**.

- Il est possible de réinitialiser le tableau **$_SESSION** avec la fonction **session_unset()**.
- Utile juste avant de détruire une sesssion (voir la diapo suivante)

PHP (modèle)

```php
session_unset() ;                    # réinitialise le tableau $_SESSION
```

- **session_destroy()** termine et supprime la session courante.
- Si on appelle **session_start()** peu de temps après avoir détruit la session, PHP réutilise parfois le même identifiant de session.
- Pour démarrer une nouvelle session et éviter ce problème il faut utiliser la fonction **session_regenerate_id()** :

PHP (modèle)

```
session_unset();                  # réinitialise le tableau $_SESSION
session_destroy();                # détruit la session courante
session_regenerate_id(TRUE); # utilise un nouvel id de session
```

On veut développer un script qui permet de retenir le nombre de fois qu'un client l'a invoqué :

- Un formulaire qui transmet une chaîne de caractères.
- Un script (l'action du formulaire) qui affiche la chaîne de caractères reçues et qui compte le nombre d'accès.

Form.html

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <title>Formulaire</title>
  </head>
  <body>
    <h1>Ajoutez des nombres</h1>
    <form action="action.php" method="get">
      Entrez un nombre ou 0 pour arrêter :<br>
      <input type="text" name="nombre">
      <br><br>
      <input type="submit" value="Ajouter">
    </form>
  </body>
</html>
```

Action.php

```php
<?php
  session_start();
  if ( ! isset($_SESSION[ "total" ]) )
    $_GET[ "total" ] = 0;
  $total = $_SESSION[ "total" ] += $_SESSION[ "nombre" ];
  if ( $_SESSION[ "nombre" ] == 0 ) {
    session_unset();
    session_destroy();
    session_regenerate_id(TRUE);
  }
?>
```

Action.php (2)

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <title>Action</title>
  </head>
  <body>
      <h2>Le total actuel est <?php echo $total; ?></h2>
      <a href="formulaire.html">Ajouter un autre nombre</a>
  </body>
</html>
```