PHP - Les Sessions

Cette fiche explore les mécanismes des sessions en PHP, en se concentrant sur leur fonctionnement sans l'utilisation de la programmation orientée objet (POO) ni de bases de données. Elle détaille les principes fondamentaux, la mise en œuvre pratique et les considérations de sécurité associées.

1 - Introduction aux Sessions

Les sessions en PHP permettent de **maintenir l'état** d'un utilisateur à travers différentes pages d'un site web.

Contrairement aux `cookies`, les données de session sont stockées sur le serveur, offrant une **sécurité accrue** et la possibilité de stocker des informations plus complexes.

Exemple :

```php
<?php
// Démarrage d'une session
session_start();

// Définition d'une variable de session
$_SESSION['username'] = 'UtilisateurConnecte';

// Affichage d'une variable de session
echo "Bonjour, " . $_SESSION['username'];
?>
```

1.1 - Démarrage d'une Session

- Le démarrage d'une session se fait avec la fonction `session_start()`.
- Cette fonction doit être appelée avant tout code HTML, car elle envoie des en-têtes HTTP.
- L'ID de session PHP (souvent nommé `PHPSESSID`) est généré côté serveur au début d'une session.
- Une fois généré, il est envoyé au navigateur du client, généralement via un cookie HTTP nommé `PHPSESSID`
- Le navigateur stocke ce cookie et le renvoie automatiquement au serveur à chaque requête suivante sur ce domaine.
- L'ID de session est une chaîne aléatoire suffisamment longue pour empêcher toute prédiction ou collision: en PHP natif, il est généré par la fonction `session_create_id()` et l'algorithme utilise une source aléatoire sécurisée.
- Il existe un lien direct : l'id de session généré sur le serveur = la valeur contenue dans le cookie PHPSESSID sur le navigateur.

1.1.1 - Gestion du Cookie de Session

- Lorsqu'une session est démarrée, PHP envoie un `cookie` au navigateur de l'utilisateur.
- Ce `cookie`, nommé en général `PHPSESSID` contient un identifiant unique (`session ID`) qui permet au serveur de retrouver les données de session associées à cet utilisateur.

Manipulation des Variables de Session

Les variables de session sont stockées dans le tableau associatif `$_SESSION`.

Ce tableau est accessible sur toutes les pages où `session_start()` a été appelée.

Exemple :

```php
<?php
session_start();

// Définition de plusieurs variables de session
$_SESSION['nom'] = 'Dupont';
$_SESSION['age'] = 30;

// Modification d'une variable de session
$_SESSION['age'] = 31;

// Suppression d'une variable de session
unset($_SESSION['nom']);
?>
```

2.1 - Ajout et Modification

Pour ajouter ou modifier une variable de session, il suffit d'assigner une valeur à une clé du tableau `$_SESSION` .

2.2 - Suppression

La fonction `unset()` permet de supprimer une variable de session spécifique. Pour détruire complètement une session, on utilise la fonction `session_destroy()` .

Fermeture d'une Session

> Il est important de fermer une session lorsque l'utilisateur a terminé sa navigation ou souhaite se déconnecter.

Exemple :

```php
<?php
session_start();

// Suppression de toutes les variables de session
// Efface toutes les variables stockées dans la session courante
$_SESSION = array();

// Destruction de la session
// Cela efface toutes les données associées à la session côté serveur
session_destroy();

// Suppression du cookie de session
// Cette étape est cruciale pour supprimer l'identifiant de session côté navigateur, notamment si vous utilisez les cookies (ce qui est
        le cas par défaut)
if (ini_get("session.use_cookies")) {
   $params = session_get_cookie_params();
   setcookie(session_name(), '', time() - 42000,
      $params["path"], $params["domain"],
      $params["secure"], $params["httponly"]
   );
}

// Redirection vers la page de connexion
header("Location: login.php");
exit();
?>
```

Sécurité des Sessions

> La sécurité des sessions est primordiale pour protéger les informations sensibles des utilisateurs.

> Voici quelques mesures à prendre en compte.

4.1 - Régénération de l'ID de Session

Il est recommandé de régénérer l'ID de session régulièrement, notamment lors de l'authentification d'un utilisateur, pour prévenir les attaques de fixation de session.

Exemple :

```php
<?php
session_start();
session_regenerate_id(true); // true pour supprimer l'ancienne session
?>
```

4.2 - Utilisation de HTTPS

L'utilisation de HTTPS est essentielle pour chiffrer la communication entre le navigateur et le serveur, empêchant ainsi l'interception des `cookies` de session.

4.3 - Stockage Sécurisé des Données

Évitez de stocker des informations sensibles directement dans les variables de session. Si nécessaire, chiffrez ces données avant de les stocker.

A RETENIR

Points essentiels :

- Les sessions stockent les données côté serveur, contrairement aux `cookies`
- `session_start()` doit être appelé avant tout code HTML
- Les données de session persistent tant que la session n'est pas détruite
- La régénération de l'ID et l'utilisation de HTTPS sont cruciales pour la sécurité
- Les sessions permettent de maintenir l'état utilisateur entre les pages

Concept	Description
`session_start()`	Fonction obligatoire pour démarrer une session
`$_SESSION`	Tableau associatif contenant toutes les variables de session
`session_destroy()`	Détruit complètement une session
`session_unset()`	Supprime toutes les variables de session
`session_id()`	Récupère l'ID de session actuel
`session_name()`	Définit ou récupère le nom de la session
`session_set_cookie_params()`	Définit les paramètres du cookie de session
`unset()`	Supprime une variable de session spécifique
`session_regenerate_id()`	Régénère l'ID de session pour la sécurité