Développement Web - Les cookies

1. Généralités

Le cookie est un petit fichier texte enregistré dans la mémoire du navigateur côté client, qui peut contenir n'importe quelle valeur.
Sa durée de vie peut être limitée mais contrairement à une variable de session, il n'est pas supprimé à la fermeture du navigateur du visiteur.

La façon dont les cookies sont stockés peut varier en fonction du navigateur et du système d'exploitation.

Dans certains navigateurs, les cookies sont stockés dans un fichier texte sur le disque dur de l'ordinateur du client. Dans d'autres navigateurs, les cookies sont stockés dans une base de données ou dans la mémoire vive de l'ordinateur.

Par exemple, dans le navigateur Google Chrome, les cookies sont stockés dans un fichier SQLite nommé "Cookies" dans le répertoire de profil de l'utilisateur. Dans le navigateur Firefox, les cookies sont stockés dans un fichier texte nommé "cookies.sqlite" dans le répertoire de profil de l'utilisateur.

Donc, même si les cookies sont stockés dans la mémoire du navigateur côté client, ils peuvent être stockés sous forme de fichier ou de base de données en fonction du navigateur et du système d'exploitation.

2. Manipulation d'un cookie en PHP

Les cookies sont accessibles par la variable globale $_COOKIES

Ajouter un cookie	setcookie("nomCookie",1, time() + 60, '/'); 1 : valeur du cookie time() + 60 : durée d'utilisation de 60 secondes, si la valeur est à 0 (valeur par défaut) le cookie expiera à la fin de la session. '/' : le cookie sera utilisable sur l'ensemble du domaine
Tester l'existence d'un cookie	if (isset($_COOKIES['nomCookie']))
Supprimer un cookie	setcookie('nomCookie', '', **time() - 1**); La date de validité étant dépassée, le cookie est supprimé.

setcookie() définit un cookie qui sera envoyé dans l'en-tête 'set-cookie' avec le reste des en-têtes. Comme pour les autres en-têtes, les cookies doivent être envoyés avant toute autre sortie. Cela vous impose d'appeler cette fonction avant toute balise <html> ou <head>.
La fonction comporte plusieurs paramètres : le nom du cookie, sa valeur, sa durée de vie, Le chemin sur le serveur sur lequel le cookie sera disponible. Si la valeur est '/', le cookie sera disponible sur l'ensemble du domaine.
D'autres paramètres existent, ils sont facultatifs car ils possèdent une valeur par défaut :
Deux sont intéressants à connaitre :

secure
Indique si le cookie doit uniquement être transmis à travers une connexion sécurisée HTTPS depuis le client. Lorsque ce paramètre vaut true, **le cookie ne sera envoyé que si la connexion est sécurisée**. Côté serveur, c'est au développeur d'envoyer ce genre de cookie uniquement sur les connexions sécurisées (par exemple, en utilisant la variable $_SERVER["HTTPS"]).

httponly
Lorsque ce paramètre vaut true, le cookie ne sera accessible que par le protocole HTTP. Cela signifie que **le cookie ne sera pas accessible via des langages de scripts, comme Javascript** ce qui réduit le risque d'exploitation par des attaques de type XSS (Cross-Site Scripting).

Une autre signature existe depuis PHP 7.3 pour créer un cookie
https://www.php.net/manual/fr/function.setcookie.php

```
setcookie(string $name, string $value = "", array $options = []): bool
```

Le tableau $options est un tableau associatif dont les clés correspondent aux différents paramètres
Cette syntaxe offre l'avantage de pouvoir préciser uniquement les paramètres nécessaires sans avoir à respecter l'ordre défini dans la première syntaxe

Créé avec cette syntaxe, la suppression d'un cookie devra utiliser cette même syntaxe

Exemple : Définition d'un cookie dont la durée de vie est de 24 heures

```
$option['expires'] = time() + 3600 * 24;
$option['path'] = '/';
$option['httponly'] = true;
$option['secure'] = true;
setcookie('visiteur', 1, $option);
```

Suppression de ce cookie

```
$option['expires'] = time() -1;
$option['path'] = '/';
$option['httponly'] = true;
$option['secure'] = true;
setcookie('visiteur', '', $option);
```

3. Comment circule un cookie entre le serveur et le client (navigateur)

Lorsqu'un cookie est créé côté serveur, il est envoyé au client dans l'en-tête HTTP de la réponse. Cela se fait en incluant un en-tête "Set-Cookie" dans la réponse HTTP

```
Set-Cookie: nom_du_cookie=valeur_du_cookie; expires=Sat, 02-Apr-2025 12:00:00 GMT;
path=/; domain=example.com; secure; HttpOnly
```

Pour indiquer que les options Secure et HttpOnly sont désactivées dans l'entête Set-Cookie, il suffit de ne pas inclure ces directives

Le navigateur reçoit ces informations et stocke le cookie conformément aux instructions données dans cet en-tête.

Lorsque le client envoie une requête vers le serveur concernant le même domaine et le même chemin, le cookie est envoyé dans l'en-tête HTTP "Cookie" de la requête si elle encore valide.

```
Cookie: nom_du_cookie1=valeur_du_cookie1; nom_du_cookie2=valeur_du_cookie2
```

Remarque : la suppression physique d'un cookie stocké dans le navigateur se produit lorsqu'une nouvelle requête est envoyée au serveur, et non immédiatement à l'expiration du cookie

4. Manipulation en Javascript côté client

Les cookies sont conservées dans la propriété cookie de l'objet document sous la forme d'une chaine de caractère de la forme "nom=valeur;expires=date;path=chemin;nom=valeur;…"

Pour créer un cookie, il faut utiliser la propriété cookie de l'objet doocument

```
document.cookie = 'nom=valeur; expires=date au format UTC; path=/'
```

Un cookie se compose de plusieurs paramètres dont le nom et la valeur sont obligatoires.

Les autres paramètres :

'expires' indique la date d'expiration du cookie. Un cookie est automatiquement supprimé lorsque la date d'expiration est atteinte. Si un cookie ne possède pas de date d'expiration, le cookie est détruit à la fin de la session (fermeture du navigateur). Pour exprimer la date dans le format attendu on utilise la méthode toUTCString() de l'objet Date.

```
let date = new Date(Date.Now() + 60 * 60 * 1000)  // ajout de ms soit 1 heure ici
'expires =' + date.toUTCString()
```

On peut remplacer le paramètre 'expires' par le paramètre 'max-age' qui indique la durée de vie du cookie en seconde à partir de sa date de création.

```
'max-age = 3600'
```

'path' indique le répertoire à partir duquel le cookie est accessible. la valeur '/' permet d'utiliser le cookie sur l'ensemble du site. Si sa valeur est nulle ou non renseignée, le cookie sera accessible à partir du l'url de la page qui l'a créé.

'domaine' précise le domaine dans lequel le cookie est accessible.

Le paramètre 'securite' devra être égal à true si le cookie est prévu pour les connexions sécurisées par https.

Pour modifier la valeur d'un cookie, il suffit de le réécrire avec le même nom et le même chemin

Pour augmenter la durée de vie d'un cookie, il faut le supprimer et le recréer avec une nouvelle durée de validite. Cela s'explique par le principe de transfert d'un cookie entre le serveur et le navigateur

Pour supprimer un cookie, il suffit de le réécrire avec le même nom et le même chemin en précisant une date d'expiration passée ou mettre une valeur négative pour l'attribut 'max-age'.
Dans ce cas, il n'est pas utile de préciser la valeur. La date d'expiration étant dépassée, le cookie ne sera tout simplement pas renvoyé vers le serveur dans l'entête 'cookie'

```
document.cookie = 'user=; path=/; expires=Thu, 01 Jan 2024 00:00:00 UTC';
```

```
document.cookie = 'user=; path=/; 'max-age = -1';
```

5. Exemples d'utilisation

5.1. Limiter la durée de vie d'une variable de session.

Une variable de session permet souvent de contrôler l'accès à une fonctionnalité. Si cette fonctionnalité est sensible (fonction d'administration par exemple), il est souhaitable d'en limiter sa durée de vie, or une session reste valable tant que l'utilisateur n'a pas mis fin à sa session.
Pour mettre fin prématurément à une session (unset) après un certain temps d'inactivité, il faut associer la variable de session à un cookie dont la durée de vie peut être définie.

Coté serveur, lors de la création de la variable de session on crée un cookie avec une durée de vie limitée.
Lorsque le navigateur, envoie une requête vers le serveur, il envoie dans l'entête 'Cookie' le cookie associé uniquement si sa date de validité n'est pas dépassée.
Côté serveur, il suffit de tester l'existence du cookie ce qui revient à dire s'il a été envoyé. Si ce n'est pas le cas, la variable de session doit être supprimée.

```
if (isset($_COOKIE["connexionActive"]))     {
    setcookie("connexionActive",1, time() + 300);
} else {
    unset($_SESSION['administrateur']);
}
```

5.2. Mise en place de la fonctionnalité 'Se souvenir de moi'

On peut utiliser un cookie pour mémoriser la connexion d'un utilisateur et la recharger automatiquement évitant ainsi le passage par la grille de connexion lors de sa prochaine visite.

Ce cookie doit contenir des informations chiffrées permettant d'identifier et d'authentifier l'utilisateur.

Pour être solide, cette empreinte doit utiliser des informations sur l'utilisateur prises dans la base de données mais aussi des informations liées au poste de travail afin d'éviter le vol de cookies.

6. Communication des cookies entre le serveur et le client

Lorsque le serveur crée un cookie, il envoie un en-tête Set-Cookie dans la réponse HTTP au navigateur. Le navigateur stocke ensuite le cookie dans son cache et le renvoie au serveur dans les en-têtes Cookie des requêtes HTTP suivantes.

Le navigateur peut accéder et modifier tous les cookies stockés dans son cache, qu'ils aient été créés côté client ou côté serveur.

Si un cookie créé côté serveur contient des informations sensibles, telles que les informations d'authentification de l'utilisateur ou les données de session, il est important d'utiliser l'attribut HttpOnly pour empêcher le code JavaScript côté client d'accéder au cookie.