Sécurisation d'application : les cookies et les sessions

Sécurisation d'application : les cookies et les sessions

Dans ce chapitre nous allons aborder :

- Ce qu'est qu'un cookie, ses paramètres et comment l'utiliser
  nous y intégrerons donc
  - le httponly
  - le protocole https
  - le paramétrage de notre serveur en https
  - le https-secure

- Ce qu'est une session, son tableau associatifs pour les variables et comment l'utiliser

Qu'est-ce qu'un cookie ?

Un cookie => petit fichier texte pouvant contenir qu'une quantité limitée de données.

Un cookie => stocké en local sur l'ordinateur de l'utilisateur. Cela signifie donc que ce dernier pourra le supprimer quand il voudra.

Un cookie => durée de vie limitée. On devra donc définir une date d'expiration.

Un cookie => souvent utilisé pour facilité l'usage du site aux utilisateurs en préenregistrant des données les concernant comme un pseudo par exemple.

Notez toutefois qu'il ne faut en aucun cas stoker des données sensibles dans un cookie.

Comment créer un cookie ?

Utiliser la fonction setcookie().

Attention avec une restriction HTTP, il faut appeler cette fonction avant tout code HTML, car les cookies doivent être envoyés avant toute sortie.

setcookie() => 7arguments. Cependant 1seul sera obligatoire à savoir le nom du cookie créé.

Syntaxe => `setcookie('myCookie', value, expire, path, domain, secure, httponly);`

Sécurisation d'application : les cookies et les sessions

Les arguments à la loupe :

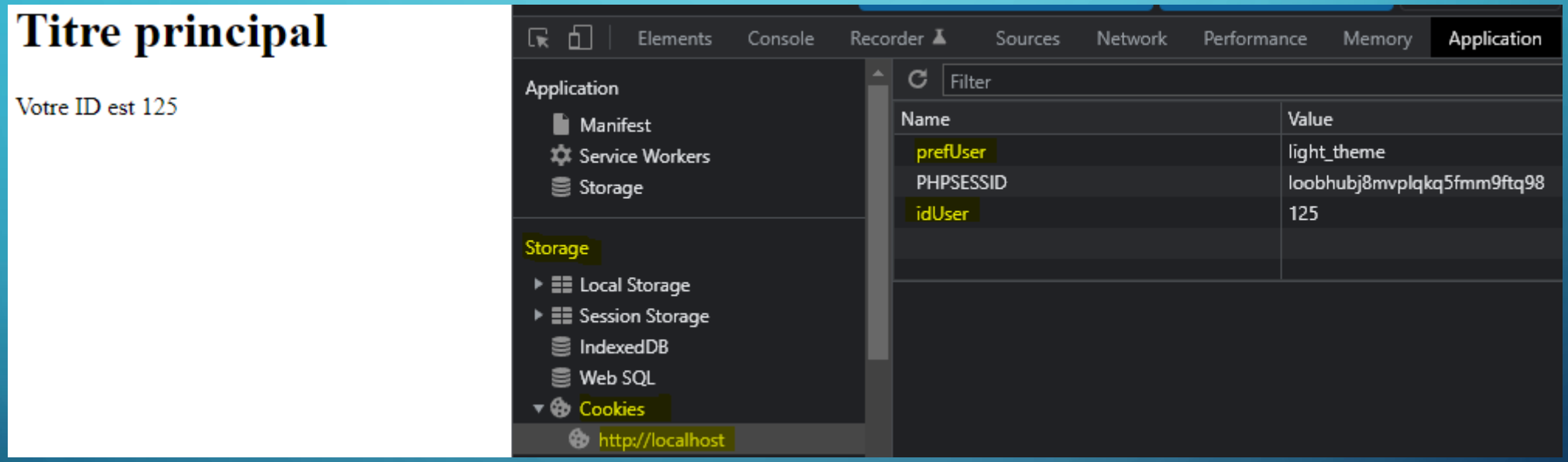
Paramètre	Signification
name	Nom du cookie soumis aux même règles que les noms de variables.
value	Valeur du cookie pouvant être un pseudo par exemple. Il faut éviter d'y stocker des données dites sensibles comme un mot de passe par exemple.
expires	Date d'expiration du cookie. Sous forme d'un timestamp(nombre de secondes écoulés depuis le 1er janvier 1970). Par défaut, si aucune valeur d'expiration n'est définie, le cookie expirera à la fermeture de la session (c'est-à-dire à la fermeture du navigateur).
path	Chemin du serveur pour lequel le cookie sera disponible. Exemple: si valeur sur '/' = cookie disponible pour l'ensemble du domaine et si valeur est '/undossier/' ne sera disponible que dans ce dossier et ses sous-dossiers.
domain	Domain ou sous-domain pour lequel le cookie est disponible.
secure	Si sa valeur est sur *true,* le cookie ne sera envoyé que si la connexion est sécurisée, c'est-à-dire en HTTPS.
httponly	Si sa valeur est sur *true*, le cookie ne sera accessible que par un protocole HTTP. Cela signifie qu'on interdit l'accès de ce cookie à des langages de scripts (tel que javascript par exemple), et permet potentiellement de se prémunir d'une attaque XSS.

Sécurisation d'application : les cookies et les sessions

```php
<?php
    //création d'un premier cookie avec l'idUser
    // qui nous permettra de l'identifier pendant toute la durée de sa navigation sur notre site
    setcookie('idUser', 125);
    //création d'un second cookie qui enregistrera les préférences de l'utilisateur
    // comme  par exemple son choix de theme entre le sombre et le clair
    // time() => fonction qui récupère la valeur du timestamp actuel
    // à laquelle on ajoute 3600 secondes * 24 à partir de sa date de création.
    // '/' => cookie accessible pour tout le domain
    // '' => Étant en local pas de domain de validité
    // 1er true => cookie accessible seulement en HTTPS
    // 2nd true => cookie accessible seulement en protocole HTTP
    setcookie('prefUser', 'light_theme', time()+3600*24, '/', '', true, true);
?>
<!DOCTYPE html>
<html lang='fr'>
    <head>
        <title>cookie and co</title>
        <meta charset="utf-8">
    </head>

    <body>
        <h1>Titre principal</h1>
        <?php
            if(isset($_COOKIE['idUser'])){
                // ici j'affiche l'idUser si le cookie idUser existe
                echo 'Votre ID est ' .$_COOKIE['idUser'];
            }
            if(isset($_COOKIE['perfUser'])){
                // ici j'affiche l'idUser si le cookie idUser existe
                echo 'Votre avez choisi le theme : ' .$_COOKIE['prefUser'];
            }
        ?>
    </body>
</html>
```

On constate que malgré la présence des cookies dans le Storage => cookie, j'affiche ce que je souhaite c'est-à-dire l'idUser pour le 1er echo, mais je n'affiche pas la préférence utilisateur du second echo.



Cela est dû au fait que notre serveur wamp (ou xampp pour certains), n'est pas configuré en HTTPS.

Pour y remédier, 2 solutions :

1_ modifier / supprimer le cookie incriminé (ici le prefUser).

2_ modifier notre serveur local (wamp ou xampp) afin que celui-ci passe en HTTPS.

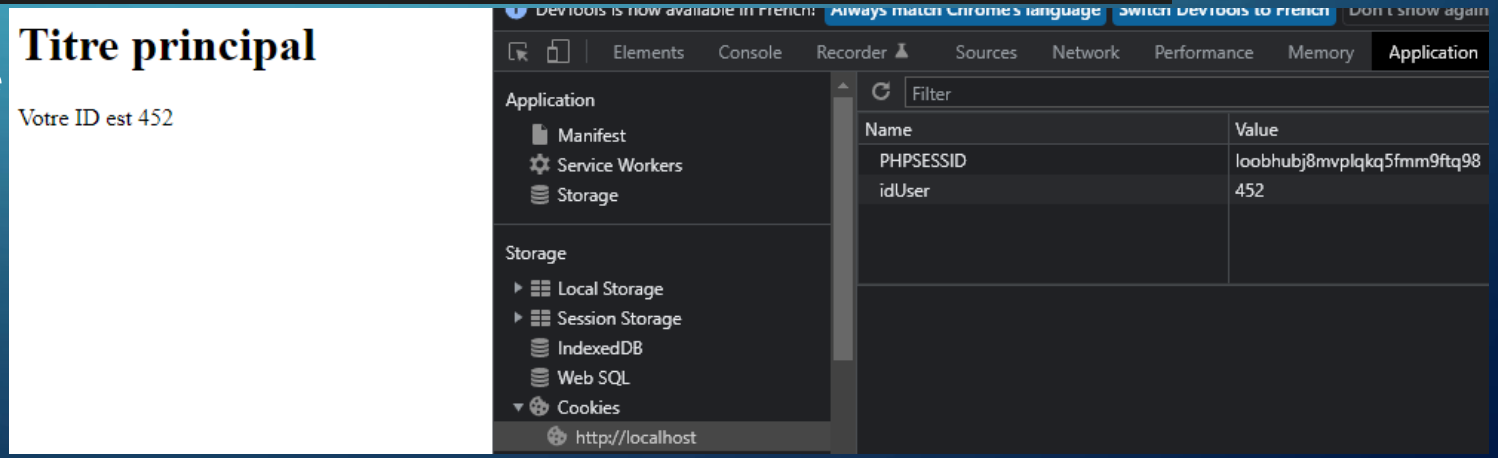
Comment modifier ou supprimer un cookie ?

Modification => setcookie('nomcookie', 'nouvelle valeur');
Suppression => setcookie('nomcookie', 'pas nécessaire', time() – 3600, '/', '', false, false);
time() – 3600 = engendre un temps dans le passé ayant pour effet la suppression du cookie.

```php
//On modifie la valeur du cookie idUser
setcookie('idUser', '75432');

//On supprime le cookie prefUser en enlevant du temps par rapport au timestamp actuel
// et en modifiant le https-secure en false tout comme le httponly
// (attention ceci est pour le bien de la démonstration seulement)
setcookie('prefUser', '', time()-3600, '/', '', false, false);
```

Le cookie 'idUser' a bien été modifié et le cookie 'prefUser a bien été supprimé.



Le protocole HTTPS qu'est-ce que c'est et comment le configurer ?

Le HTTPS est l'extension du protocole HTTP.
HTTPS => HyperText Transfert Protocole Secure (protocole de transfert hypertexte sécurisé en français).

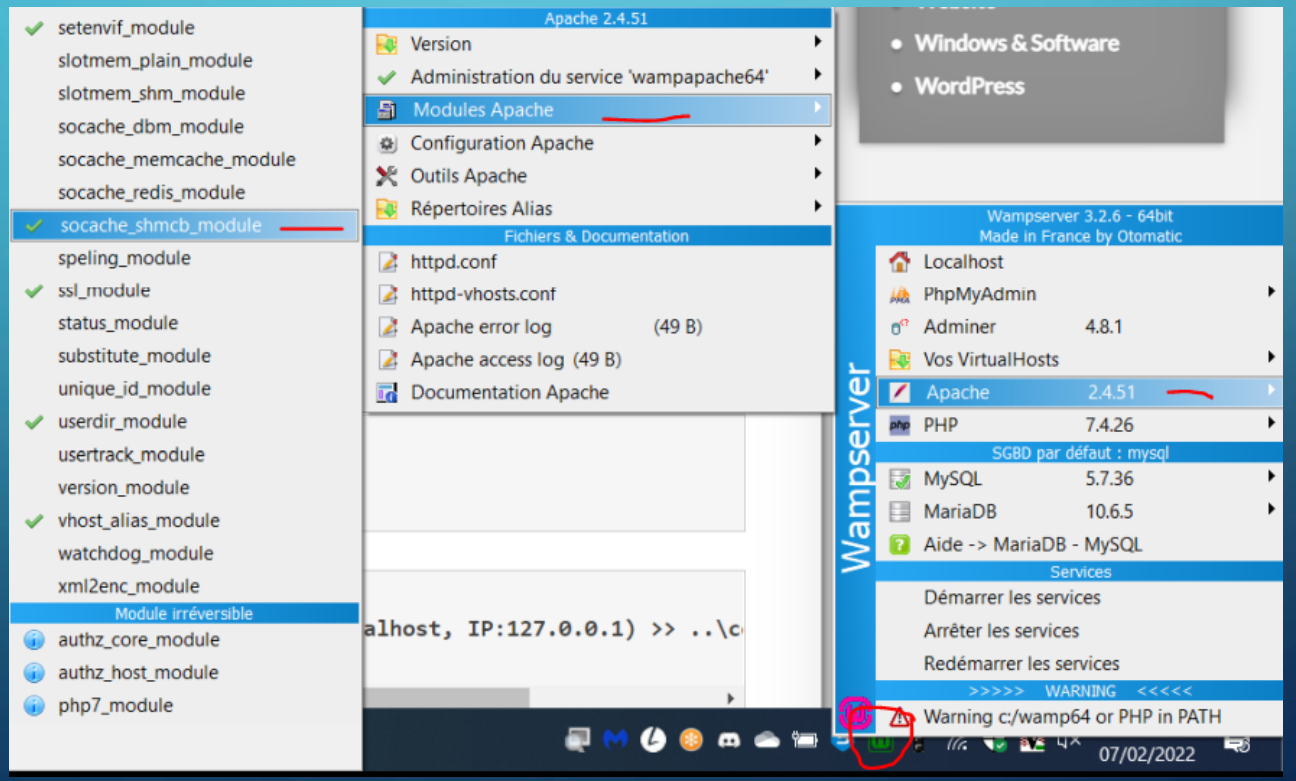
S => couche de sécurité en SSL ou TLS.
SSL => Secure Socket Layer (couche de socket sécurisé) est le prédécesseur du TLS.
TLS => Transport Layer Security (Sécurité de la couche de transport).
SSL et TLS sont des protocoles de sécurisation des échanges par réseau informatique.
Suivant un mode client-serveur, ils permettent les objectifs suivants :

- Authentification du serveur.
- Confidentialité des données échangées (chiffrage de la session).
- Intégrité des données échangées.
- Et parfois, authentification du client (plus souvent assurée par l'application).

Mise en place sur wamp du protocole HTTPS en local sur wamp :

(pour xampp veuillez suivre ce tuto)

- Commencez par cliquer sur l'icone de wamp, puis allez sur Apache => Modules apache => socache_shmcb_module



Sécurisation d'application : les cookies et les sessions

- Ouvrez une invite de commande en mode administrateur et tapez => cd c:\wamp\bin\apache\apacheX.X.XX\bin (la partie en rouge peut être légèrement différente selon votre installation.

```
C:\Users\jonathangruson\.openssl>cd C:\wamp64\bin\apache\apache2.4.51\bin
```

- Tapez la ligne suivante dans votre invite de commande => (echo [my_req_ext] && echo subjectAltName = DNS:localhost, DNS:*.localhost, IP:127.0.0.1) >> ..\conf\openssl.cnf

- Continuez sur l'invite de commande avec les3 lignes suivantes :

```
openssl genrsa -aes256 -passout pass:ABCDE -out mypriv.key 2048 && openssl rsa -in mypriv.key -passin pass:ABCDE -out mypriv.key

openssl req -new -x509 -nodes -sha1 -key mypriv.key -out mycert.crt -days 44444 -config ..\conf\openssl.cnf -extensions my_req_ext -subj "/C=US/ST=Distributed/L=Cloud/O=Cluster/CN=localhost"

copy mypriv.key ..\conf\mypriv.key /y && copy mycert.crt ..\conf\mycert.crt /y
```

Sécurisation d'application : les cookies et les sessions

- Rendez-vous dans le dossier c:\wamp\bin\apache\apacheX.X.XX\conf et ouvrez le fichier httpd.conf avec un éditeur de texte (de preference notepad++)
- Recherchez le ligne Define APACHE_DIR et ajoutez Define SRVROOT ${APACHE_DIR}/
- Trouvez et décommentez Include conf/extra/httpd-ssl.conf
- Puis allez dans le dossier c:\wamp\bin\apache\apacheX.X.XX\conf\extra pour y ouvrir le fichier httpd-ssl.conf.
- Dans ce fichier recherchez la ligne <VirtualHost _default_:443>

Sécurisation d'application : les cookies et les sessions

- Sous la ligne que vous venez de trouver, modifiez les lignes suivantes :

  DocumentRoot "${INSTALL_DIR}/www"
  ServerName localhost:443
  ...
  ...
  SSLCertificateFile "${SRVROOT}/conf/mycert.crt"
  ...
  SSLCertificateKeyFile "${SRVROOT}/conf/mypriv.key“

- Enfin sauvegardez les modifications effectuées

- Dans votre invite de commande saisissez httpd –t pour verifier si votre syntaxe est correcte.

```
C:\wamp64\bin\apache\apache2.4.51\bin>httpd -t
Syntax OK
```

- Et pour terminer, redémarrez votre wampserveur et tester l'adresse https://localhost.

Qu'est-ce qu'une session ?

En php, une session est une façon de stocker des données différentes pour chaque utilisateur par l'intermédiaire d'un identifiant unique de session.
Contrairement aux cookies qui sont stockés sur la navigateur et donc côté client, les informations de session seront-elles côté serveur, rendant ainsi les sessions plus sûres.

Attention, une session démarrera dès l'instant où la fonction session_start() sera appelée et se terminera par défaut lors de la fermeture de la fenêtre courante du navigateur.

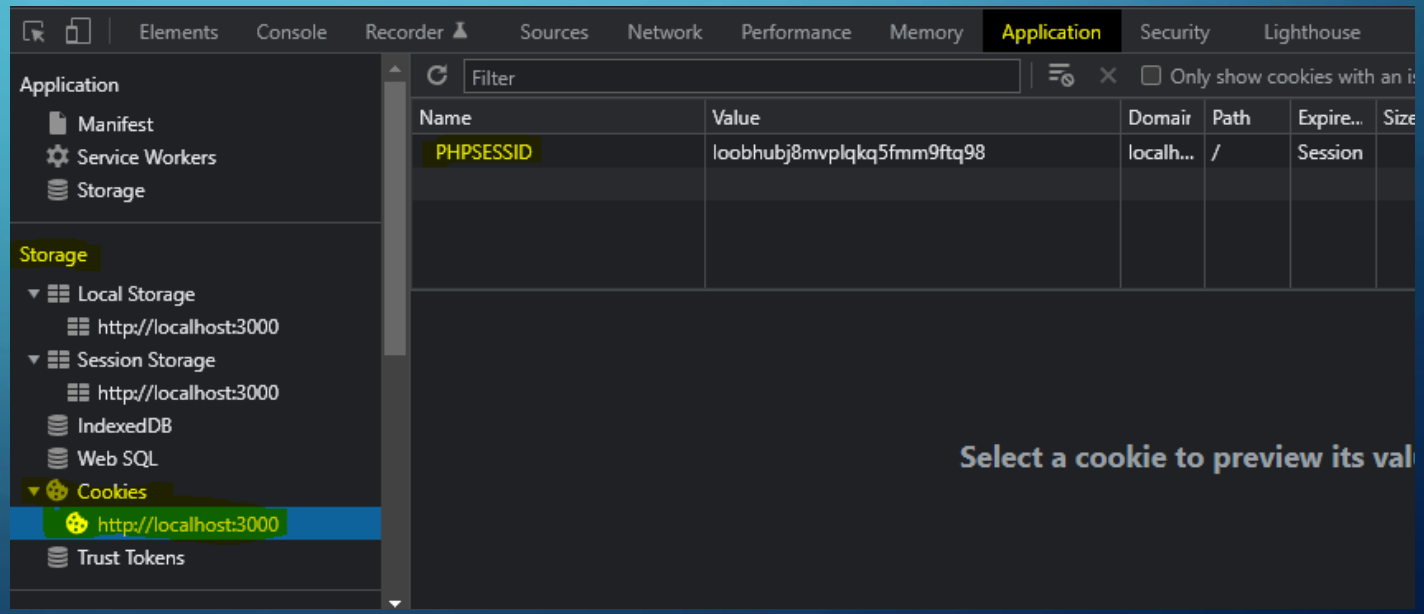
Pour accéder à ces informations de session, nous utiliserons la superglobale $_SESSION.

Démarrer une session ?

Pour démarrer une session => session_start().

Appel de cette fonction sur chaque page où on a besoin d'accéder aux variables de session.

Une fois cette session démarrée, elle va donc générer un identifiant de session unique et en général celui-ci sera envoyé au navigateur sous une forme de cookie nommé 'PHPSESSID'.



Sécurisation d'application : les cookies et les sessions

Exemple d'une page qui affiche l'id de session.


```php
<?php
    //On démarre une nouvelle session
    session_start();
    /*On utilise session_id() pour récupérer l'id de session s'il existe.
     *Si l'id de session n'existe  pas, session_id() renvoie une chaine
     *de caractères vide*/
    $id_session = session_id();
?>
<!DOCTYPE html>
<html lang='fr'>
    <head>
        <title>Exemple</title>
        <meta charset="utf-8">
    </head>
    <body>
        <h1>page exemple</h1>
        <?php
            if($id_session){
                echo 'ID de session (récupéré via session_id()) : <br>'
                .$id_session. '<br>';
            }
            echo '<br><br>';
            if(isset($_COOKIE['PHPSESSID'])){
                echo 'ID de session (récupéré via $_COOKIE) : <br>'
                .$_COOKIE['PHPSESSID'];
            }
        ?>
    </body>
</html>
```

Créer et récupérer des variables de session ?

$_SESSION = superglobale
Donc tout comme $_GET, $_POST c'est un tableau associatif.

Ceci signifie donc que pour créer une variable de session je ferais :

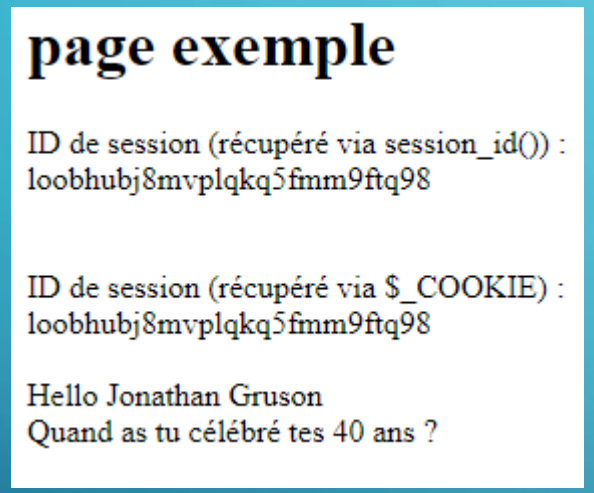
$_SESSION['mavariable'] = ''la valeur que je souhaite lui donné'';

Dans l'exemple précédent, je vais donc créer des variables comme ceci :

```php
<?php
    //On démarre une nouvelle session
    session_start();
    /*On utilise session_id() pour récupérer l'id de session s'il existe.
     *Si l'id de session n'existe  pas, session_id() renvoie une chaine
     *de caractères vide*/
    $id_session = session_id();

    // je définis des variables
    $_SESSION['prenom'] = "Jonathan";
    $_SESSION['nom'] = "Gruson";
    $_SESSION['age'] = 40;
?>
```

Et je vais faire les utiliser pour les afficher :




```php
<body>
    <h1>page exemple</h1>
    <?php
        if($id_session){
            echo 'ID de session (récupéré via session_id()) : <br>'
            .$id_session. '<br>';
        }
        echo '<br><br>';
        if(isset($_COOKIE['PHPSESSID'])){
            echo 'ID de session (récupéré via $_COOKIE) : <br>'
            .$_COOKIE['PHPSESSID'];
        }
        echo '<br><br>';
        echo 'Hello '.$_SESSION['prenom'].' '.$_SESSION['nom'].'
            <br>Quand as tu célébré tes '.$_SESSION['age'].' ans ?';
    ?>
</body>
```

Finir une session et supprimer les variables de session ?

Par défaut, php terminera une session d'un utilisateur lorsque ce dernier fermera la fenêtre correspondante.

Mais parfois, il serait préférable de le faire avant et donc utiliser la fonction session_destroy(), qui détruira ainsi toutes les données correspondant à la session en cours et également la fonction session_unset(), qui quant à elle supprimera toutes les variables d'une session.

Attention toutefois, session_destroy() ne supprimera que le fichier de session dans lequel se trouve les informations de session.
Cela signifie que les variables globales de cette session dans le tableau $_SESSION et le cookie de session PHPSESSID seront toujours présent.
Pour les supprimer définitement en utilisant la fonction setcookie() et en définissant une date d'expiration.