

Introduction à PHP

Cookies et Sessions

`monnerat@u-pec.fr`

19 mai 2019

IUT de Fontainebleau

Cookies

Sessions

Cookies

Cookie

Les cookies permettent de stocker du côté client des données associées à un visiteur particulier.

Pour permettre par exemple :



- d'identifier de manière unique un visiteur.
- de préremplir les données d'un formulaire le concernant pour lui éviter de les resaisir.
- de connaître ses derniers achats.

D'abord conçus par Netscape, ils sont maintenant pris en charge par tous les navigateurs, et intégré au protocole HTTP.

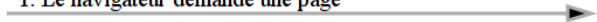
- Petit fichier texte stocké par le navigateur sur l'ordinateur du visiteur, à la demande du serveur.
- Le serveur envoie un cookie avec la page, stocké coté client.
- Il demande ainsi au navigateur du client de le lui renvoyer à chaque échange ; (le serveur peut ainsi accéder à l'information du cookie, la modifier...)

Les navigateurs peuvent refuser.

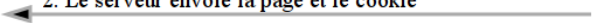
Navigateur Web

Serveur Web

1. Le navigateur demande une page



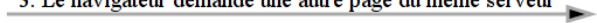
2. Le serveur envoie la page et le cookie



Le cookie

Hello World!

3. Le navigateur demande une autre page du même serveur



Le cookie

Cookies et HTTP

La demande de cookie (serveur ↔ client) se fait dans les en-têtes HTTP :

```
Set-Cookie : NOM=VALEUR; domain=NOM DE DOMAINE; expires=DATE
```

Envoi des cookies au serveur (client ↔ serveur) lorsque le client fait une requête à un serveur, les cookies pour le domaine et le chemin sont envoyés dans les en-têtes HTTP sous la forme

```
Cookie : NOM1=VALEUR1; NOM2=VALEUR2; ...
```

Remarques

- La taille maximale d'un cookie est de 4 ko.
- PHP permet de les gérer.

Exemple avec google

The screenshot shows a web browser window with the Google France homepage. The address bar shows the URL `https://www.google.fr`. Below the browser window, the Network Developer Tools console is open, displaying the details of a GET request to `www.google.fr`. The console shows the response status as 200 OK and the size as 28.1 KB. The 'Cookies' tab is selected, showing a list of cookies sent by the server.

URL	Statut	Domaine	Poids	Remote IP	Chronologie
GET www.google.	200 OK	google.fr	28,1 KB	173.194.34.55:443	68ms

Cookies envoyés								
Nom	Contenu	Valeur brute	Hôte	Taille	Chemin	Expire	HttpOnly	Sécurité
⊕ PREF	ID=b228e65ID=b228e652a92489f9:U=f...IG=1:S=Uy4VHmX-kvzi8Mmx			105 B				
⊕ NID	67=QkfZpU:67=QkfZpU53FAgrfZ-jN8K...5JL5IPLPqAvrApPHPg8Wl0w			173 B				
⊕ SID	DQAAAOIAAAQDQAAAOIAAADQjUMdya7y1WZ...4qvbTB0xYcF_gxOKAvkpecc			334 B				
⊕ HSID	A7v3WtqtKMA7v3WtqtKMAC9GX3b			21 B				
⊕ SSID	AfNAPc3XbcAfNAPc3XbcKdD046Y			21 B				
⊕ APISID	XQZEb-UjFalXQZEb-UjFalPjzWA-4OxhqOekyrPdCi1			40 B				
⊕ CAPSID	FrArvMm_CFrArvMm_Cx0DfX7 /ArdAMwLidW9-hFYaT			41 B				

La fonction setcookie

La fonction

```
bool setcookie ( string $name [, string $value  
    [, int $expire = 0 [, string $path [, string $domain  
    [, bool $secure = false [, bool $httponly = false ]]]]] )
```

est la fonction qui permet de gérer les cookies. `setcookie()` définit un cookie qui sera envoyé avec le reste des en-têtes. Les cookies doivent passer avant le corps de la requête (c'est une restriction HTTP, pas de PHP). Cela vous impose d'appeler cette fonction avant toute balise html ou écriture depuis php dans le "flux html".

En cas d'échec, `setcookie()` retournera `FALSE`. Si `setcookie()` réussit, elle retournera `TRUE`. Cela n'indique pas si le client accepte ou pas le cookie. La RFC 6265 est la référence pour l'interprétation des paramètres.

Paramètres de setcookie

- `name` : Le nom du cookie. 'cookienome' est appelé via `$_COOKIE['cookienome']`
- `value` : La valeur du cookie cette valeur est retrouvé en utilisant `$_COOKIE['cookienome']`
- `expire` : Le temps après lequel le cookie expire. timestamp Unix, donc, ce sera un nombre de secondes depuis l'époque Unix (1 Janvier 1970). En d'autres mots, vous devriez fixer cette valeur à l'aide de la fonction `time()` et en y ajoutant le nombre de secondes après lequel on veut que le cookie expire. Vous pouvez utiliser aussi `mktime()`

Exemple

`time()+60*60*24*30` fera expirer le cookie dans 30 jours. Si vous ne spécifiez pas ce paramètre, le cookie expirera à la fin de la session (lorsque le navigateur sera fermé).

Paramètres de setcookie

- `path` : le chemin sur le serveur sur lequel le cookie sera disponible

Exemple

Si la valeur est `'/'`, le cookie sera disponible sur l'ensemble du domaine `domain`. Si la valeur est `'/foo/'`, le cookie sera uniquement disponible dans le répertoire `/foo/` ainsi que tous ces sous-répertoires comme `/foo/bar/` du domaine `domain`. La valeur par défaut est le répertoire courant où le cookie a été défini.

- `domain` : le domaine où le cookie est disponible.

Exemple

Pour rendre le cookie disponible sur tous les sous-domaines de `example.com`, vous devez mettre la valeur `'.example.com'`. Le point (`.`) n'est pas requis mais est nécessaire pour la compatibilité avec encore plus de navigateurs.

Paramètres de `setcookie`

- `secure` : indique si le cookie doit uniquement être transmis à travers une connexion sécurisée HTTPS. Lorsqu'il est positionné à TRUE, le cookie ne sera positionné uniquement si la connexion sécurisée existe. La valeur par défaut est FALSE.
- `httponly` : lorsque ce paramètre vaut TRUE, le cookie n'est pas accessible depuis les langages de scripts coté client, comme Javascript.

Tous les arguments sauf `name` (nom) sont optionnels.

Remarque

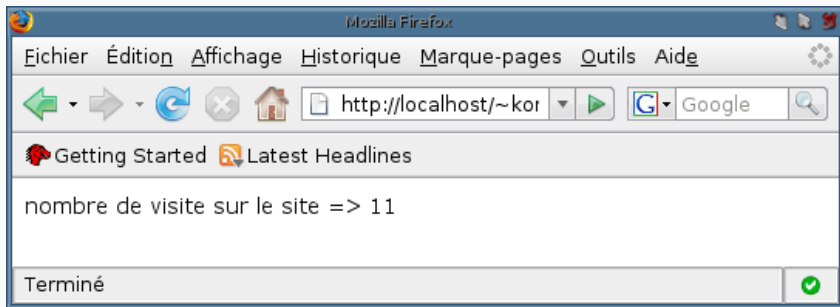
Pour effacer un cookie, il faut utiliser `setcookie` avec une date antérieure à la date courante. Le plus simple :

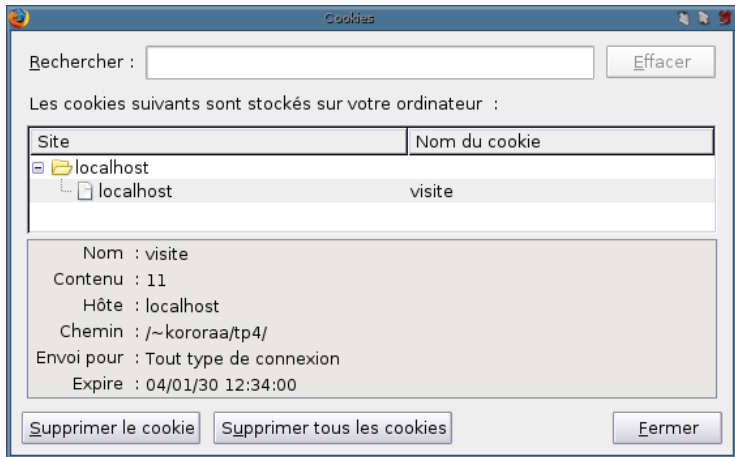
```
setcookie("moncookie", "", 1);
```

Exemple

```
$visite = $_COOKIE['visite'] ?? 0; // PHP7
$visite++;
/*
if(!isset($_COOKIE['visite']))
    $visite=1;
else
    $visite=$_COOKIE['visite']+1;
*/
setcookie('visite',$visite,mktime(12,34,00,04,01,2030));
$message="nombre de visite sur le site => $visite";
?>
<html>
  <head>
  </head>
  <body>
    <?php echo $message;?>
  </body>
```

- La première fois, le cookie est créé, avec la valeur 1.
- Par la suite, il est envoyé au serveur, récupéré à l'aide de `$_COOKIE['visite']`, incrémenté, et renvoyé au client.





Remarque

Pour stocker une information plus complexe qu'un nombre ou une chaîne de caractères, par exemple un tableau, utilisez les fonctions :

- `serialize` pour transformer le tableau en une chaîne de caractères, stockable dans un cookie.
- `unserialize` pour réaliser l'opération inverse.

```
<?php
$moi=array (
    "nom"=>"monnerat",
    "bureau"=>114
);
echo serialize($moi);
?>
```

affiche la chaîne :

```
a:2:{s:3:"nom";s:8:"monnerat";s:6:"bureau";i:114;}
```


Sessions

Dans la partie précédente, nous avons vu que les cookies permettent de stocker de l'information coté client. Inconvénients

- Taille des données est limitée.
- Le client peut les modifier.

Pour les données sensibles \Rightarrow Sessions

Session

Une session est un ensemble d'informations associé avec un utilisateur particulier et conservé (sur le serveur) tout au long de son interaction avec les différentes pages d'un site Web.

- Une session attribue à un visiteur un identifiant unique, stocké dans un cookie.
- A chaque fois qu'il revient en donnant cette identifiant, PHP peut récupérer toutes les informations sauvegardées pour cette utilisateur.
- Les données sont sauvegardées sur le serveur.
- Contrairement aux cookies, les sessions ont une durée de vie limitée, en particulier à la fermeture du navigateur.

Les sessions sont un peu plus adaptées au stockage de données confidentielles, par exemple :

- Contrôle d'accès après authentification.
- Gestion d'un panier d'achat.
- Formulaire en plusieurs parties. (on stocke les données déjà saisies)
- Disposer d'un cache pour certaines ressources très coûteuses. (requête sur une base)

Les sessions s'appuient sur la notion de cookie, donc ne résolvent pas complètement le problème de la sécurité.

1. `session_start() /* initialisation*/`

- Cette fonction initialise la session. PHP essaie de lire l'identifiant fourni par l'utilisateur, va chercher le fichier correspondant et met à disposition l'information correspondante dans la variable (super globale) `$_SESSION`
- Si aucun identifiant de session n'est reçu, PHP en crée un unique aléatoire, l'envoie au visiteur et crée le fichier pour stocker l'information de session.

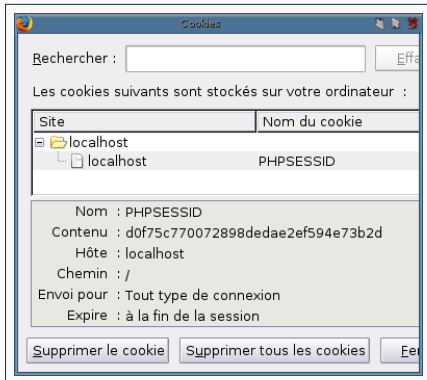
2. Par la suite, l'information est disponible en lecture ou écriture dans le tableau associatif `$_SESSION`.

3. `session_destroy() /* suppression de la session */`

- La fonction détruit le fichier de données associé sur le serveur.
- Après `session_destroy`, dans le script en cours, on peut effacer le tableau `$_SESSION=array()`.

Remarque : Le numéro de session est généré dans l'entête http. Il ne faut donc mettre aucun code html (pas même une ligne blanche) avant l'appel à la fonction `session_start()`

- L'identifiant de session est stocké dans un cookie nommé par défaut PHPSESSID
- Ce cookie contient l'identifiant assigné à la session.
- S'il n'existe pas, il est créé de manière aléatoire puis envoyé au navigateur par cookie (sans date d'expiration).



C'est le fichier `php.ini` qui gère les sessions

```
; Whether to use cookies.
```

```
session.use_cookies = 1
```

```
;session.cookie_secure =
```

```
; This option enables administrators to make their users invulnerable  
; attacks which involve passing session ids in URLs; defaults to
```

```
session.use_only_cookies = 0
```

```
; Name of the session (used as cookie name).
```

```
session.name = PHPSESSID
```

```
; Initialize session on request startup.
```

```
session.auto_start = 0
```

```
; Lifetime in seconds of cookie or, if 0, until browser is restarted
```

```
session.cookie_lifetime = 0
```


Contrôle d'accès avec une session

1. Un formulaire HTML demande un nom utilisateur et un mot de passe.
2. Le script qui traite le formulaire vérifie que nom d'utilisateur et mot de passe fournis sont corrects (en les comparant à des valeurs stockées dans une base de données, par exemple) :

si c'est le cas, il crée et affecte une nouvelle variable de session :

```
$_SESSION['valid_user']=1;
```

ensuite il redirige l'utilisateur vers une des pages protégées.

Si le login n'est pas correct le script redirige vers le formulaire de login.

Toutes les pages protégées commencent par vérifier que l'utilisateur a bien été authentifié :

```
<?php
include dirname(__FILE__).'/verification.php';
?>
```

en incluant le fichier

```
if(!isset($_SESSION['valid_user'])
    || $_SESSION['valid_user']!=1) {
    header('Location:authentification.html');
    exit;
}
```

Le site doit prévoir au moins une page de déconnexion qui fait appel à `session_destroy`

- Dans la base de données, on ne stocke **jamais** les mots de passes en clair.
- On stocke une empreinte (hash) du mot de passe.

Il est recommandé d'utiliser la fonction `password_hash`

```
echo password_hash("toto",PASSWORD_DEFAULT);
```

```
$2y$10$Ymz.xrvVrKqgGQWm6r.ux.e9rBaf0sChRjGLDinAoW6UnePPOT/Tu
```

La fonction utilise (entre autres) des paramètres par défaut (on peut les fixer)

- un coût de calcul
- un sel

Pour vérifier l'empreinte d'un mot de passe, on utilisera la fonction `password_verify`

```
<?php
$hash = '$2y$10$Ymz.xrvVrKqgGQWm6r.ux.e9rBaf0sChRjGLDinAoW6UnePPOT/Tu'

if (password_verify('toto', $hash)) {
    echo 'Le mot de passe est valide !';
} else {
    echo 'Le mot de passe est invalide.';
}
?>
```

Vérification

Le mot de passe est considéré comme correct si son empreinte (calculée) est identique à celle stockée dans la base de données.

Un exemple

- Le jeu de la devinette : le retour. Il s'agit d'essayer de deviner un nombre entre 0 et 100 en cinq tentatives.

Eichier Édition Affichage Historique Marque-pages Tuile Outils Aide

Devinette

localhost Google

Les plus visités Getting Started Latest Headlines PostBac

Jeu de la devinette

Entrez un entier envoyer

Il vous reste 5 tentatives.

session = 587b9aeq68qrogbd18m6a21rn1

- Deux variables de sessions `nbadeviner` et `nbt`.

```
[root@portable_denis tmp]# cat sess_587b9aeq68qrogbd18m6a21rn1  
nbt|i:4;nbadeviner|i:78;[root@portable_denis tmp]#
```

```
<?php
session_start();
//echo "session = ".session_id()."<br /><br />";//.session_name
if (!isset($_SESSION['nbt'])) {
    $_SESSION['nbt']=5;
    $_SESSION['nbadeviner']=rand()%100;
    $action="";
    // echo $_SESSION['nbadeviner'];
}else{
    $_SESSION['nbt']--;
    if ($_SESSION['nbt']==0){ session_destroy();}
    $nb=$_POST['nb'];

    if ($_SESSION['nbadeviner']>$nb) $action='+';
    else if ($_SESSION['nbadeviner']<$nb) $action='-';
    else $action='=';
}
?>
```

```

<html>
  <head>
    <LINK href="../../tp3/style.css" rel="stylesheet" type="text/css">
    <title>Jeu de la devinette</title>
  </head>
  <body>
    <?php
    if($action=="") {
      echo "bravo, vous avez trouve";
      session_destroy();
    }
    if ($_SESSION['nbt']==0&&$action!="") echo "vous avez perdu.
    Le nombre etait " .$_SESSION['nbadeviner'];
    ?>
    <form action="" method="post">
      <fieldset>
        <legend> Jeu de la devinette </legend>
        <label> entrez un entier <?echo $action;?></label>
        <input type="text" name='nb'>
        Il vous reste <?echo $_SESSION['nbt'];?> tentatives.
      </fieldset>
      <input type="submit" value=<?php if ($action==""||$_SESSION['nbt']==0)
      echo "rejouer" ;else echo "envoyer";?>
    </form>
  </body>
</html>

```