

L i a m T A R D I E U

www.evogue.fr





Sommaire

➤	Sommaire	2
➤	Cookies	3
	Introduction.....	3
	Fonctionnement.....	3
	Sécurité.....	3
	Création et modification	4
	Lecture.....	4
	Suppression	4
	Stocker des tableaux dans des cookies.....	5
	Exemple concret avec les cookies	5
	Observer les cookies	7
➤	Sessions	8
	Introduction.....	8
	Fonctionnement.....	8
	Initialisation.....	9
	Création, modification et lecture	9
	Suppression	9
	Exemple	10
➤	Communication avec la base de données	11
	Introduction.....	11
	La classe Mysqli.....	11
	La classe Mysqli_Result.....	13
	Connexion à la base de données	13
	Effectuer une requête	14
	Gestion des erreurs.....	14
	Exploiter les données en PHP	14
	Traitement d'un enregistrement.....	15
	Traitement de plusieurs enregistrements	15
	Traitement des enregistrements et affichage visuel.....	16
	Choix du connecteur de base de données	17
➤	Sécurité.....	18
	Faibles de Sécurité XSS.....	18
	Faibles de sécurité Injection SQL	19
	Moyen de protection	20
	Autres Faibles de sécurité	21
	Restriction d'accès sur les dossiers.....	22



Cookies

INTRODUCTION

Les cookies servent à enregistrer et retenir des informations concernant le système de l'internaute. Le plus généralement, ils sont utilisés pour :

- stocker des informations saisies (ou des choix faits) par l'internaute (ex. : des préférences)
- effectuer du traitement statistique

FONCTIONNEMENT

Un cookie est donc un fichier texte placé sur le système de l'internaute. Il se présente sous une forme similaire à celle-ci :

```
nom ; valeur ; domaine ; chemin ; sécurité ;  
expiration
```

L'entité nom + valeur représente le cookie lui-même. Le domaine (*domain*) et la date d'expiration (*expires*) délimitent le champ de validité du cookie. C'est la même chose pour le chemin (*path*), mais cet attribut est optionnel. Le paramètre de connexion (*secure*) permet de limiter le cookie à une connexion sécurisée (SHTTP par exemple).

La valeur d'un cookie est toujours une chaîne.

Les spécifications définissent les limites suivantes pour les cookies : 20 cookies par domaine et 4ko max. (environ 4000 caractères) par cookie. Il est généralement possible de dépasser ces limites, les navigateurs l'acceptant plutôt bien... mais il est tout de même conseillé de rester en dessous de celles-ci.

SECURITE

Sécurité de l'internaute

Les cookies font parfois peur aux internautes ; il n'y a pourtant pas lieu de s'inquiéter. D'abord parce que les cookies sont créés par le serveur web, et que celui-ci ne dispose pas d'autres informations sur vous que celles que vous voulez bien lui donner. Ensuite parce que seul le serveur qui a créé un cookie est capable de le lire.

Sécurité du script

Un cookie étant un simple fichier texte dans la mémoire du système, il est tout à fait possible à un internaute averti de modifier les cookies sur son système.

Il est donc évident que les cookies ne doivent pas servir à stocker n'importe quel type de données. Notamment, ne stockez jamais dans un cookie les montants des articles d'un panier d'achat. Sinon, il serait possible à l'internaute de modifier le prix de ses achats avant de passer commande.

CREATION ET MODIFICATION

Pour créer un cookie, on utilise la fonction `setcookie()` :

```
$date=date("U")+(7*24*60*60);  
setcookie("nomducookie","valeurducookie",$date);
```

Pour modifier un cookie, on utilise la même fonction. En effet, si le cookie existe, sa valeur est remplacée par la nouvelle, et s'il n'existe pas encore, il est alors créé.

LECTURE

Pour lire (et utiliser) un cookie, on utilise la superglobale `$_COOKIE` :

```
if(!isset($_COOKIE["compteur"])){  
    setcookie("compteur","1");  
}  
else{  
    $i=$_COOKIE["compteur"];  
    $i++;  
    setcookie("compteur",$i);  
}
```

SUPPRESSION

Pour supprimer un cookie, il n'existe là encore aucune fonction spécifique...

Il suffit de recréer le cookie sans valeur :

```
setcookie("nom du cookie");
```

Ou bien, on peut lui attribuer une date d'expiration antérieure à la date courante.

```
setcookie("nom du cookie","",1);  
// la valeur 1 correspond au 1er janvier 1970 à minuit et 1 seconde)
```

Attention, il ne faut pas confondre le cookie et la superglobale `$_COOKIE` : supprimer le cookie ne supprime pas la superglobale. Inversement, les modifications de la superglobale n'affectent pas le cookie.

STOCKER DES TABLEAUX DANS DES COOKIES

Comme il est impossible de stocker directement un tableau dans un cookie (un cookie ne peut stocker qu'une seule valeur), il va falloir trouver un autre moyen. Nous utiliserons une fonction de linéarisation pour transformer le tableau en chaîne, et une fonction de "dé-linéarisation" pour retransformer la chaîne en tableau lors de la lecture du cookie.

Fonction de linéarisation :

```
$chaîne = serialize ($tableau) ;
```





Fonction de dé-linéarisation :

```
$tableau = unserialize ($chaîne) ;
```

```
if(!isset($_COOKIE["unTableau"])){  
    $tab = array("pomme", "poire", "banane", "fraise");  
    $valeur = serialize($tab);  
    setcookie("unTableau", $valeur);  
}  
else{  
    print_r(unserialize(stripslashes($_COOKIE["unTableau"])));  
}
```

EXEMPLE CONCRET AVEC LES COOKIES

Nous allons créer un site multilingues autour de 4 pays : France, Espagne, Angleterre, Italie. Il serait fastidieux pour les visiteurs, à chaque connexion sur le site, d'avoir à reparamétrer la langue dans laquelle s'affiche le contenu du site. C'est pour cela que nous allons sauvegarder cette information dans un cookie.

Votre langue:
 France
 Espagne
 Angleterre
 Italie

Ciao
Si sta attualmente visitando il sito in Italiano
Infatti, il cookie consente il backup di mantenere la lingua che si usa il sito per visite future.
Presto.

langue.php

```
<?php
if(isset($_GET['pays']))
{ $pays=$_GET['pays']; }
elseif(isset($_COOKIE['pays']))
{ $pays=$_COOKIE['pays']; }
else{ $pays='fr'; }

$temps = 365*24*3600; // 1an
setCookie("pays",$pays,time()+$temps);

switch($pays)
{
    case 'fr': print '<p>Bonjour</p>'; break;
    case 'es': print '<p>¡Hola</p>'; break;
    case 'an': print '<p>Hello</p>'; break;
    case 'it': print '<p>Ciao p>'; break;
}
?>
<a href="?pays=fr"> France </a><br /><br />
<a href="?pays=es"> Espagne </a><br /><br />
<a href="?pays=an">Angleterre </a><br /><br />
<a href="?pays=it">Italie </a><br /><br />
```

Nous passons l'argument de la langue « it », « fr », « es », « an », en GET dans l'url. Cet argument est conservé dans la variable \$pays.

Si le cookie pays existe déjà nous le récupérons dans la variable \$pays.

Sinon, si le cookie n'existe pas et que le visiteur n'as pas fait de choix de langue, la variable \$pays est déclarée à « fr » par défaut et, quoi qu'il en soit, nous sauvegardons le cookie sur le pc de l'internaute avec cette valeur.

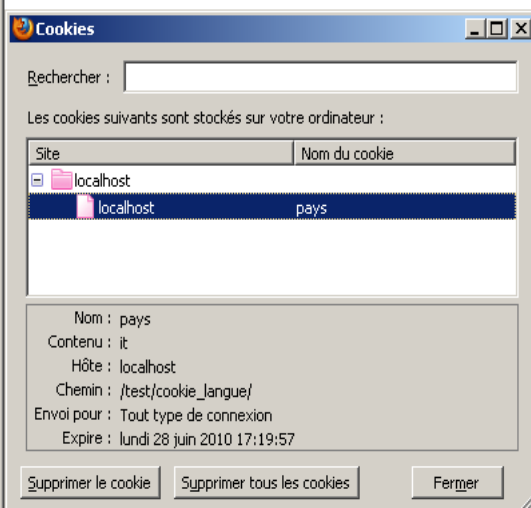
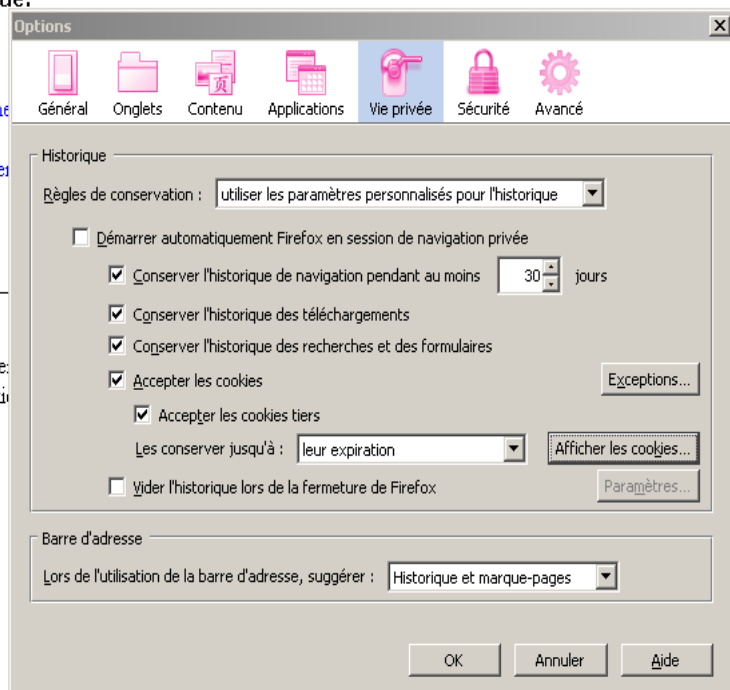
OBSERVER LES COOKIES

Sur Mozilla Firefox, il faut se rendre dans : Outils -> Options -> Vie privée -> Afficher les cookies

Votre langue:



Ciao
Si sta attualmente
Infatti, il cookie conser
Presto.

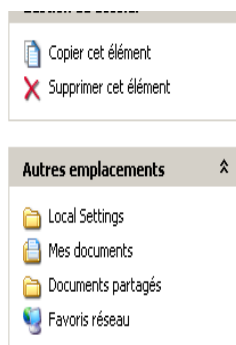


Sur Internet Explorer, il faut aller dans : Outils -> Options internet -> [dans la zone historique de navigation] Paramètres -> Afficher les fichiers.

Votre langue:

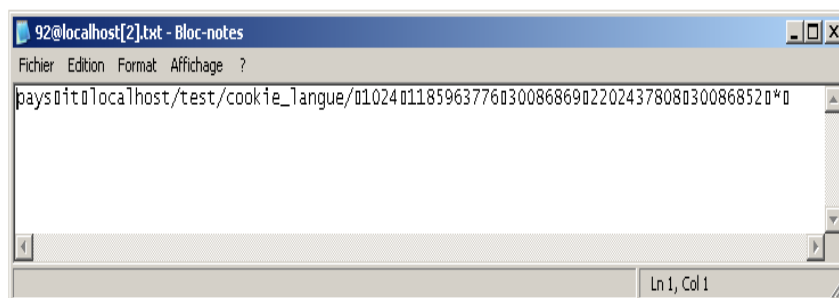


Ciao
Si sta attualmente visi
Infatti, il cookie conser
Presto.



cookie_langue/ Cookie:92@localhost/test/cookie_langue/

Document texte 1 Ko





Sessions

INTRODUCTION

Puisque les cookies connaissent certaines limites (utilisation restreinte à des chaînes, nombre et poids limités, problème de sécurité), PHP nous propose une autre façon de conserver des informations d'une page à l'autre : les sessions.

En effet, les sessions ont une importance capitale car sans elle (par exemple) un internaute perdrait systématiquement les articles mis dans son panier lors de sa navigation ou perdrait la connexion (en tant que membre) établit avec le site quand il naviguerait d'une page à une autre.

Par rapport aux cookies, les sessions ont l'avantage de pouvoir stocker des variables de tous types (chaînes, nombres, tableaux, objets...), et de ne pas avoir de limite quantitative. Par contre, elles ont une durée de vie très limitée par rapport aux cookies (généralement quelques heures...).

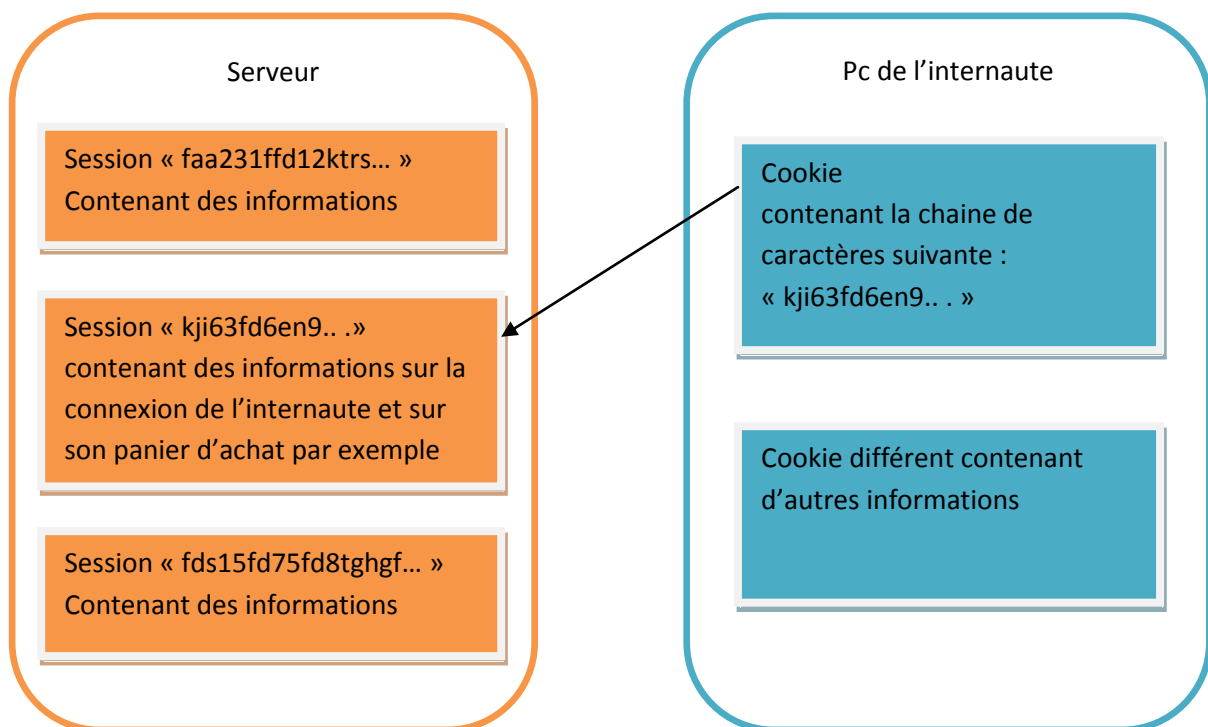
FONCTIONNEMENT

Une session est un fichier stocké sur le serveur (nous pouvons en général les observer dans le dossier nommé /tmp/ dans l'arborescence du serveur).

Pour chaque utilisateur du script, la session renvoie un identifiant unique (c'est cet identifiant qui identifie également le fichier du serveur).

Ce fichier-session peut stocker tous types de données. Chaque donnée est automatiquement "linéarisée" au stockage, et automatiquement "délinéarisée" à la lecture.

Quand une session est créée, cela crée automatiquement un cookie sur le pc de l'utilisateur, nous appelons cela le cookie de session (PHPSESSID) :



INITIALISATION

Tous les fichiers PHP utilisant des sessions doivent d'abord initialiser les sessions, c'est le rôle de la fonction :

```
session_start() ;
```

Cette initialisation doit se faire sur tous les fichiers utilisant la session, pas seulement sur le premier. Lors de l'appel de cette fonction prédéfinie PHP, un fichier de session est automatiquement créé (vide dans un premier temps) sur le serveur ainsi qu'un cookie de session sur le pc de l'internaute (le cookie contiendra le même id que l'id de session généré). Par exemple, c'est de cette manière que le site sait si vous êtes bien connecté. Il s'agit de l'ouverture du fichier session.

CREATION, MODIFICATION ET LECTURE

Une fois la session initialisée, toutes les variables qu'elle contient sont disponibles depuis la superglobale \$_SESSION :

```
session_start();
if(!isset($_SESSION["compteur"])){
    $_SESSION["compteur"]=1;
}
else{
    $_SESSION["compteur"]++;
}
```

SUPPRESSION

Pour détruire une session, nous utilisons la fonction :

```
session_destroy() ;
```

Attention, il faut penser à initialiser la session avant de la détruire.

Pour la « vider » de son contenu, nous utiliserons la fonction :

```
unset($_SESSION["compteur"]) ;
```

Certains sites (surtout les banques) détruisent automatiquement la session à l'issue d'une période d'inactivité sur leurs pages.

EXEMPLE

```
session_start();
$_SESSION["pseudo"] = "Julien";
$_SESSION["mdp"] = "COTTET";
echo "<hr />n°1 : ";
print_r($_SESSION);

unset($_SESSION['mdp']);
echo "<hr />n°2 : ";
print_r($_SESSION);

session_destroy();
echo "<hr />n°3 : ";
print_r($_SESSION);
```

Renseignement sur la session

```
session_start()
print session_id();
```

`session_id()` est utilisée pour récupérer ou définir l'identifiant de session pour la session courante. La méthode suivante remplace l'identifiant de session courant par un nouveau :

```
session_regenerate_id()
```

A retenir

Sous WampServer, nous pouvons observer les sessions dans le dossier `/tmp/`.

Quand une session est créée sur le serveur, un cookie est également créé sur le pc de l'internaute.

Dans ce cas, on parle alors de cookie de sessions. (PHPSESSID).

Nous pouvons retrouver l'identifiant de la session dans son contenu, c'est ce qui nous permet de faire le lien entre le fichier sur le pc de l'internaute et le fichier de session sauvegardé sur le serveur.



Communication avec la base de données

INTRODUCTION

Il est possible de mélanger les langages PHP et SQL, nous parlons alors d'accouplement des technologies : PHP/MYSQL.

De cette manière, le contenu des sites ne se trouve plus forcément dans le corps des fichiers mais plutôt dans une base de données, si nous ajoutons à cela une partie Front-Office (FO) et une partie Back-Office (BO), nous parlons alors de site dynamique.

Avec cette avancée, les mises à jour ne se font plus directement dans le corps du fichier mais plutôt dans la base de données en passant par une interface proposée par la partie BackOffice et les répercussions sur le Front-Office sont immédiates puisque le BackOffice modifie la base de données et que le FrontOffice se construit et s'alimente avec ce qu'il trouve dans cette même base de données.

LA CLASSE MYSQLI

L'extension MYSQLI (abréviation pour MySQL Improved en anglais, c'est-à-dire, en français, MySQL Amélioré) est un pilote qui permet d'interfacer des programmes écrits dans le langage de programmation PHP avec les bases de données MySQL.

La classe MYSQLI est une classe prédéfinie, elle est native et chargée automatiquement par PHP. MYSQLI et PDO sont des extensions définissant l'interface pour accéder à une base de données depuis PHP.

L'extension MYSQLI a la particularité de pouvoir s'utiliser librement dans un contexte de code procédural ou de code orienté objet.

Pour plus d'informations, il faut se reporter au chapitre sur les classes et les objets dans le support concerné.

L'objet MYSQLI représente une connexion entre PHP et un serveur de base de données.

Propriétés et méthodes

`mysqli::$affected_rows` — Retourne le nombre de lignes affectées par la dernière opération MySQL

`mysqli::$autocommit` — Active ou désactive le mode auto-commit

`mysqli::$change_user` — Change l'utilisateur de la connexion spécifiée

`mysqli::$character_set_name` — Retourne le jeu de caractères courant pour la connexion

`mysqli::$client_info` — Récupère les informations du client MySQL

`mysqli::$client_version` — Retourne la version du client MySQL sous la forme d'une chaîne de caractères

`mysqli::$close` — Ferme une connexion

`mysqli::$commit` — Valide la transaction courante

`mysqli::$connect_errno` — Retourne le code d'erreur de la connexion MySQL

`mysqli::$connect_error` — Retourne le message d'erreur de connexion MySQL

`mysqli::$__construct` — Ouvre une connexion à un serveur MySQL

`mysqli::$debug` — Effectue des actions de débogage

`mysqli::$disable_reads_from_master` — Désactive la lecture depuis le maître

`mysqli::$dump_debug_info` — Écrit les informations de débogage dans les logs

`mysqli::$errno` — Retourne le dernier code d'erreur produit

`mysqli::$error_list` — Retourne une liste d'erreurs depuis la dernière commande exécutée

`mysqli::$error` — Retourne une chaîne décrivant la dernière erreur

`mysqli::$field_count` — Retourne le nombre de colonnes pour la dernière requête

`mysqli::get_charset` — Retourne un objet représentant le jeu de caractères

`mysqli::get_client_info` — Récupère des informations sur le client MySQL

`mysqli_get_client_stats` — Retourne des statistiques sur le client, par processus

`mysqli_get_client_version` — Retourne la version du client MySQL sous forme de chaîne de caractères

`mysqli::get_connection_stats` — Retourne des statistiques sur la connexion

`mysqli::$host_info` — Retourne une chaîne contenant le type de connexion utilisée

`mysqli::$protocol_version` — Retourne la version du protocole MySQL utilisé

`mysqli::$server_info` — Retourne la version du serveur MySQL

`mysqli::$server_version` — Retourne un entier représentant la version du serveur MySQL

`mysqli::get_warnings` — Lit le résultat de SHOW WARNINGS

`mysqli::$info` — Retourne des informations à propos de la dernière requête exécutée

`mysqli::init` — Initialise MySQLi et retourne une ressource à utiliser avec `mysqli_real_connect()`

`mysqli::$insert_id` — Retourne l'identifiant automatiquement généré par la dernière requête

`mysqli::kill` — Demande au serveur de terminer un thread MySQL

`mysqli::more_results` — Vérifie s'il y a d'autres jeux de résultats MySQL disponibles

`mysqli::multi_query` — Exécute une requête MySQL multiple

`mysqli::next_result` — Prépare le prochain résultat d'une requête multiple

`mysqli::options` — Définit les options

`mysqli::ping` — Ping la connexion au serveur et reconnecte si elle n'existe plus

`mysqli::poll` — Vérifie l'état de la connexion

`mysqli::prepare` — Prépare une requête SQL pour l'exécution

`mysqli::query` — Exécute une requête sur la base de données

`mysqli::real_connect` — Ouvre une connexion à un serveur MySQL

`mysqli::real_escape_string` — Protège les caractères spéciaux d'une chaîne pour l'utiliser dans une requête SQL, en prenant en compte le jeu de caractères courant de la connexion

`mysqli::real_query` — Exécute une requête SQL

`mysqli::reap_async_query` — Lit un résultat pour une requête asynchrone

`mysqli::refresh` — Rafraîchit

`mysqli::rollback` — Annule la transaction courante

`mysqli::rpl_query_type` — Retourne le type de requête RPL

`mysqli::select_db` — Sélectionne une base de données par défaut pour les requêtes

`mysqli::send_query` — Envoie la requête et retourne

`mysqli::set_charset` — Définit le jeu de caractères par défaut du client

`mysqli::set_local_infile_default` — Rétablit le gestionnaire par défaut pour la commande LOAD LOCAL INFILE

`mysqli::set_local_infile_handler` — Définit une fonction de rappel pour la commande LOAD DATA LOCAL INFILE

`mysqli::$sqlstate` — Retourne l'erreur SQLSTATE de la dernière opération MySQL

`mysqli::ssl_set` — Utilisée pour établir une connexion sécurisée avec SSL

`mysqli::stat` — Obtient le statut courant du système

`mysqli::stmt_init` — Initialise une commande MySQL

`mysqli::store_result` — Transfère un jeu de résultats à partir de la dernière requête

`mysqli::$thread_id` — Retourne l'identifiant du thread pour la connexion courante

`mysqli::thread_safe` — Indique si le support des threads est activé ou pas

`mysqli::use_result` — Initialise la récupération d'un jeu de résultats

`mysqli::$warning_count` — Retourne le nombre d'avertissements générés par la dernière requête

LA CLASSE MYSQLI_RESULT

Représente une requête, une fois exécutée, le jeu de résultats associé.

Propriétés et méthodes

`mysqli_result::$current_field` — Récupère la position courante d'un champ dans un pointeur de résultats

`mysqli_result::data_seek` — Déplace le pointeur interne de résultats

`mysqli_result::fetch_all` — Lit toutes les lignes de résultats dans un tableau

`mysqli_result::fetch_array` — Retourne une ligne de résultat sous la forme d'un tableau associatif, d'un tableau indexé, ou les deux

`mysqli_result::fetch_assoc` — Récupère une ligne de résultats sous forme de tableau associatif

`mysqli_result::fetch_field_direct` — Récupère les métadonnées d'un champ unique

`mysqli_result::fetch_field` — Retourne le prochain champ dans le jeu de résultats

`mysqli_result::fetch_fields` — Retourne un tableau d'objets représentant les champs dans le résultat

`mysqli_result::fetch_object` — Retourne la ligne courante d'un jeu de résultats sous forme d'objet

`mysqli_result::fetch_row` — Récupère une ligne de résultats sous forme de tableau indexé

`mysqli_result::$field_count` — Récupère le nombre de champs dans un résultat

`mysqli_result::field_seek` — Déplace le pointeur de résultats sur le champ spécifié

`mysqli_result::free` — Libère la mémoire associée à un résultat

`mysqli_result::$lengths` — Retourne la longueur des colonnes de la ligne courante du jeu de résultats

`mysqli_result::$num_rows` — Retourne le nombre de lignes dans un résultat

CONNEXION A LA BASE DE DONNEES

Nous ne reviendrons pas ici sur le SQL des requêtes. Pour la suite et dans nos exemples, nous reprendrons la base de données nommée "tic_entreprise" étudiée lors du cours.

Avant de tenter une connexion au serveur MySQL et à la base de données, il est indispensable de s'assurer que l'on dispose bien de l'adresse du serveur, du nom d'utilisateur et du mot de passe. Sous Wamp, l'adresse du serveur est "localhost", l'utilisateur "root", et le mot de passe "" (une chaîne vide). Sous Mamp, le mot de passe est : root.

Etablir la connexion avec le serveur

La connexion au serveur s'effectue de la manière suivante :

```
$mysqli = new mysqli("localhost", "root", "", "tic_entreprise");
if ($mysqli->connect_error)
{
    die('Un problème est survenu lors de la tentative de connexion à la BDD : ' . $mysqli->connect_error);
}
```

`$mysqli` représente un objet (une instance de classe `MYSQLI` ayant à présent une référence).

Avant d'effectuer une requête il est indispensable d'effectuer la connexion au SGBD et de choisir une base de données existante.

Pour éviter d'avoir à reproduire le script précédent sur toutes les pages utilisant la base de données, il peut s'avérer profitable de gérer la connexion au serveur MySQL depuis une inclusion de fichier.

EFFECTUER UNE REQUETE

Requête de sélection sur la table des employes.

```
$resultat = $mysqli->query("SELECT * FROM employes  
echo $resultat->num_rows;
```

- Pour exécuter une requête, nous utilisons la méthode `query()` de l'objet `MYSQLI` :
- L'attribut `num_rows` permet de connaître le nombre d'enregistrements retournés par la requête.
- La variable `$resultat` représente un objet issue de la classe `MYSQLI_RESULT`.

GESTION DES ERREURS

Exécuter une requête via l'objet `MYSQLI` peut se faire dans un bloc `try/catch`.

Il est également possible de gérer les erreurs via l'attribut (variable) : `mysqli->error`.

Exemple :

```
$mysqli->query("insert into employes (id_employes, prenom, nom,  
sexe, service, date_embauche, salaire, id_secteur) values (1100,  
'julien', 'cottet', 'm', 'secretaire', '1980-12-17', 1070, 10)");  
  
echo $mysqli->error;  
  
echo $mysqli->affected_rows;
```

- Dans cet exemple, la méthode `query()` est utilisée pour effectuer une requête d'insertion sur la table `employes`.
- L'attribut `error` est appelé pour afficher d'éventuelles erreurs sur la requête qui auraient pu compromettre l'insertion.
- L'attribut `affected_rows` est appelé pour afficher le nombre d'enregistrements affectés par l'insertion.

EXPLOITER LES DONNEES EN PHP

Pour les requêtes de d'insertion, de modification, de suppression ou encore de remplacement, l'appel de la méthode `query()` sur l'objet `MYSQLI` est suffisante. En effet, la méthode `query()` se contente de donner des instructions au serveur.

Dans le cas d'une requête de sélection, `MySQL` retourne les résultats. Il faut alors demander à `PHP` de les traiter. Il existe pour cela plusieurs fonctions de lecture de résultat de requête.

Ces fonctions utilisent un "pointeur". En effet, la requête répond enregistrement par enregistrement (ligne par ligne). Le pointeur est la référence à la ligne courante (ligne pointée). Ainsi, lors de la lecture du résultat de la requête, la ligne pointée est retournée et le pointeur avance à la ligne suivante jusqu'à la fin de la requête.

fetch_array

Retourne une ligne de résultats MySQL sous la forme d'un tableau associatif et d'un tableau indexé. Cette syntaxe est au final avantageusement remplacée par `fetch_row` pour un tableau indexé, et par `fetch_assoc` pour un tableau associatif.

fetch_row

Retourne une ligne de résultats MySQL sous la forme d'un tableau indexé (numériquement).

fetch_assoc

Retourne une ligne de résultats MySQL sous la forme d'un tableau associatif. Cette solution est souvent préconisée.

fetch_object

Retourne une ligne de résultats MySQL sous la forme d'un objet avec comme attributs les champs de la table sur laquelle la requête a été effectuée.

TRAITEMENT D'UN ENREGISTREMENT

Exemple (pour une ligne de résultats)

```
$resultat = $mysqli->query("SELECT * FROM employes where  
id_employes=7369");  
$employe = $resultat->fetch_assoc();  
print "<pre>"; print_r($employe); print "</pre>";  
print $employe['prenom'];
```

Dans cette requête, l'employé est sélectionné par son id et comme l'id est unique, nous savons qu'il n'y aura qu'un seul résultat. Pour cette raison, nous n'avons pas besoin de mettre en place de structure itérative (communément appelée : boucle).

TRAITEMENT DE PLUSIEURS ENREGISTREMENTS

Exemple (pour plusieurs lignes de résultats)

```
$resultat = $mysqli->query("SELECT * FROM employes");  
while($employes = $resultat->fetch_assoc())  
{  
    print "<pre>"; print_r($employes); print "</pre>";  
}
```

Il est fréquent que les requêtes renvoient plusieurs lignes de résultats. Nous utilisons généralement une structure itérative pour les récupérer. Cette syntaxe permet de traiter les données ligne par ligne au fur et à mesure de l'exécution de la boucle `while`, ou de tout récupérer dans un tableau multidimensionnel.

TRAITEMENT DES ENREGISTREMENTS ET AFFICHAGE VISUEL

Voici un exemple permettant de récupérer des informations d'une table (SQL) et de l'afficher sous forme de tableau (HTML).

```
$resultat = $mysqli->query("SELECT * FROM employes");
$nbcol = $resultat->field_count;

echo "<table style='border-color:red' border=10> <tr>";
for ($i=0; $i < $nbcol; $i++)
{
    $colonne = $resultat->fetch_field();
    echo '<th>' . $colonne->name . '</th>';
}
echo "</tr>";

while ($ligne = $resultat->fetch_assoc())
{
    echo '<tr>';
    foreach ($ligne as $indice => $information)
    {
        echo '<td>' . $information . '</td>';
    }
    echo '</tr>';
}
echo '</table>';
```

Nous commençons par effectuer une requête à l'aide de la méthode `query()` sur l'objet `MYSQLI` et nous récupérons le résultat via l'objet `MYSQLI_RESULT`.

Ensuite nous récupérons le nombre de champs (colonne) de la table employés via une propriété de l'objet `mysqli_result` : `field_count`. De cette manière, nous saurons combien de colonnes nous devons recréer en tableaux HTML (soit autant que dans la table SQL pour une reproduction visuelle exacte).

Nous écrivons la table html via les balises adéquates.

Nous faisons tourner une première boucle `while` pour parcourir tous les champs et obtenir des informations via la méthode `fetch_field()` afin d'écrire les en-têtes de notre tableau HTML sur la première ligne (balise `tr`).

Nous mettons en place une boucle `while` pour parcourir chaque enregistrement (soit 1 ligne par enregistrement) de la table SQL.

Il est donc important d'écrire une nouvelle ligne de notre tableau HTML (balise `tr`) à chaque tour de la boucle `while`.

Une boucle `foreach` est imbriquée à l'intérieur de la boucle `while` pour faire sortir chaque information sur chaque enregistrement.

Quand la boucle `while` se positionne sur l'employé n°1, la boucle `foreach` va parcourir toutes ses informations : *voir page suivante*

Fonctionnement de la boucle foreach imbriquée dans la boucle while

- While : Employé n°1
 - Foreach : information n°1
 - Foreach : information n°2
 - Foreach : information n°3

- While : Employé n°2
 - Foreach : information n°1
 - Foreach : information n°2
 - Foreach : information n°3

- While : Employé n°3
 - Foreach : information n°1
 - Foreach : information n°2
 - Foreach : information n°3

- While : Employé n°4
 - Foreach : information n°1
 - Foreach : information n°2
 - Foreach : information n°3

- While : Employé n°5
 - Foreach : information n°1
 - Foreach : information n°2
 - Foreach : information n°3

Chaque tour de la boucle while entraine plusieurs tours de la boucle foreach.

CHOIX DU CONNECTEUR DE BASE DE DONNEES

PHP 5 propose plusieurs connecteurs capables de se connecter à une base de données MySQL. Le premier et le plus courant reste le driver MySQL de base. C'est le plus simple à utiliser et donc le plus populaire. Il existe également l'extension MySQLi (MySQL improved) qui n'est autre qu'un driver amélioré de l'extension MySQL de base et qui a la particularité d'être orienté objet. Avec MySQLi, le développeur manipule directement un objet de type MySQLi alors qu'avec le driver MySQL de base il doit fonctionner par appel de procédure et fonctions.

Enfin, le dernier connecteur est l'extension PDO qui signifie PHP Data Objects. Cette extension permet de se connecter à de multiples bases de données à condition que les pilotes pour chaque système d'information soient installés sur le serveur Web. PDO a été intégré avec PHP 5 et a le principal avantage de faire bénéficier le développeur de certaines fonctionnalités de la base de données en fonction de son pilote. Au même titre que MySQLi, PDO est une interface orientée objet, ce qui est beaucoup plus pratique à utiliser lorsque nous sommes à l'aise avec de la programmation orientée objet.

Au fil du temps, les fonctions `mysql_*` (ou plus généralement l'extension `MYSQL`) pourraient disparaître au profit de PDO et de `MYSQLI`. Il faut donc préconiser leur utilisation si l'environnement le permet.



FAILLES DE SECURITE XSS

Introduction

Le cross-site scripting, abrégé XSS, est un type de faille de sécurité des sites Web, que l'on trouve typiquement dans les applications Web qui peuvent être utilisées par un attaquant pour provoquer un comportement du site Web différent de celui désiré par le créateur de la page (redirection vers un site, vol d'informations, etc.). Il est abrégé XSS pour ne pas être confondu avec le CSS (feuilles de style), X étant une abréviation commune pour « cross » (croix) en anglais.

Détection

La détection de la présence d'une faille XSS peut se faire par exemple en entrant un script Javascript dans un champ de formulaire ou dans une URL :

```
<script type="text/javascript">alert('bonjour')</script>
```

Si une boîte de dialogue apparaît, nous pouvons en conclure que l'application Web est sensible aux attaques de type XSS.

Les risques

L'exploitation d'une faille de type XSS permettrait à un intrus de réaliser les opérations suivantes :

- Redirection (parfois de manière transparente) de l'utilisateur.
- Vol d'informations, par exemple sessions et cookies.
- Actions sur le site faillible, à l'insu de la victime et sous son identité (envois de messages, suppression de données, etc.)
- Plantage de la page (boucle infinie d'alertes par exemple), et souvent du navigateur.
- Etc.

Une faille de type XSS est à l'origine de la propagation des virus Samy sur MySpace en 2005 ainsi que de Yamanner sur Yahoo! Mail en 2006.

FAILLES DE SECURITE INJECTION SQL

Introduction

Une injection SQL est un type d'exploitation d'une faille de sécurité d'une application interagissant avec une base de données, en injectant une requête SQL non prévue par le système et pouvant compromettre sa sécurité.

Cas d'étude

Imaginons un site internet vulnérable qui aurait pour but de connecter un utilisateur en fonction de son pseudo et de son mot de passe enregistrés en base.

L'injection serait d'injecter un « morceau de code » afin de détourner le sens de la requête initialement prévue.

Une attaque de ce type est effectuée, la plupart du temps, dans le but de récupérer un mot de passe ou de s'identifier sur le site.

Sur une page contenant un formulaire d'identification utilisateur avec les champs pseudo et mot de passe, une personne malintentionnée pourrait envoyer le bout de code suivant :

pseudo: ' OR 1='1

mdp: ' OR 1='1

Ce qui transformerait la requête initialement prévue :

```
select * from utilisateurs where pseudo='julien' and mdp='soleil';
```

Vers la requête :

```
select * from utilisateurs where pseudo='' OR 1='1' and mdp='' OR 1='1'
```

Cette requête détournée permettrait certainement de s'identifier car la requête est vraie si un utilisateur "" (chaîne vide) existe OU si 1=1. Comme il est évident que 1 est égal à 1, elle serait vraie. Nous ne sommes pas obligés de nous arrêter là, pour peu que nous nous procurons le nom de la table ou de la base, il serait possible de détruire ces dernières, de stopper le serveur, et bien autre chose encore...

Cependant, nous ne développerons pas plus ces attaques, ce chapitre à pour but de présenter les failles de sécurité (les points faibles d'un site) de manière à les prendre en considération et y remédier ; ce chapitre n'a pas pour but d'ouvrir la porte au piratage.

MOYEN DE PROTECTION

Ces attaques peuvent être évitées de plusieurs façons :

- Utiliser des procédures stockées, à la place du SQL dynamique. Les données entrées par l'utilisateur sont alors transmises comme paramètres, sans risque d'injection.
- Vérifier de manière précise et exhaustive l'ensemble des données venant de l'utilisateur. Nous pouvons, par exemple, utiliser une expression rationnelle afin de valider qu'une donnée entrée par l'utilisateur est bien de la forme souhaitée.
- Utiliser des comptes utilisateurs SQL à accès limité (en lecture-seule) quand cela est possible.
- Utiliser des requêtes SQL paramétrées (requêtes à trous envoyées au serveur SQL, serveur à qui l'on envoie par la suite les paramètres qui boucheront les trous), ainsi c'est le SGBD qui se charge d'échapper les caractères selon le type des paramètres.
- Réaliser un contrôle en amont en bloquant systématiquement les données d'URL ou de Formulaire contenant des valeurs telles que : EXEC, SELECT, INSERT, DROP, CREATE, ALTER, UPDATE, etc.
- Utiliser, en conjonction avec `mysqli_real_escape_string`, la fonction `sprintf()`.

Dans notre exemple, il serait bon de faire appel à la fonction prédéfinie : `htmlspecialchars()`.
`htmlspecialchars` convertit tous les caractères éligibles en entités HTML.

AUTRES FAILLES DE SECURITE

Découverte

D'autres failles de sécurité existent telles que le Cross-site cooking, le Cross-site request forgery, il est important de traquer les failles de sécurité de son site internet et de tenter de provoquer des bugs afin de pouvoir y remédier.

cross-site cooking

Le cross-site cooking est un type d'exploit des navigateurs Web, dans lequel un site Web crée un cookie dans le domaine d'un autre site, ce qui n'est normalement pas réalisable. En effet, lorsqu'un navigateur accepte un cookie provenant d'un serveur, il accepte en même temps des informations qui lui permettent d'identifier le serveur et de lui associer le cookie, afin que seul le site web correspondant ait accès au cookie. De même que la faille cross-site scripting, le cross-site cooking repose donc sur une interaction entre deux sites Web.

Cette faille permet notamment l'usurpation d'identité au moyen d'attaque par session fixation.

Cross-Site Request Forgery

Les attaques de type Cross-Site Request Forgery (abrégées CSRF prononcées *sea-surfing* ou parfois XSRF) utilisent l'utilisateur comme déclencheur, celui-ci devient complice sans en être conscient. L'attaque étant actionnée par l'utilisateur, un grand nombre de systèmes d'authentification est contourné. Le nombre de sites se protégeant contre ce type d'attaque est encore marginal, faute de communication sur ce type d'attaque probablement.

Exemple :

Supposons que Bob soit l'administrateur d'un forum et qu'il soit connecté à celui-ci par un système de sessions. Alice est un membre de ce même forum, elle veut supprimer un des messages du forum. Comme elle n'a pas les droits nécessaires avec son compte, elle utilise celui de Bob grâce à une attaque de type CSRF.

- * Alice arrive à connaître le lien qui permet de supprimer le message en question.
- * Alice envoie un message à Bob contenant une pseudo-image à afficher (qui est en fait un script).

L'URL de l'image est le lien vers le script permettant de supprimer le message désiré.

* Bob lit le message d'Alice, son navigateur tente de récupérer le contenu de l'image. En faisant cela, le navigateur actionne le lien et supprime le message, il récupère une page web comme contenu pour l'image. Ne reconnaissant pas le type d'image associée, il n'affiche pas d'image et Bob ne sait pas qu'Alice vient de lui faire supprimer un message contre son gré.

D'autres failles de sécurité existent, il est important de les étudier.

RESTRICTION D'ACCES SUR LES DOSSIERS

htaccess

Lors de la création d'un site internet, nous sommes souvent amenés à créer une zone d'administration où l'accès est limité... il est vivement préférable que seuls les propriétaires du site aient accès à cette partie étant donné qu'à partir de cet espace on y gère le site avec un maximum d'options. Exemple pour une boutique: gestion des commandes, gestion des paiements, gestion des produits, prix, stock, suppression des produits, etc.

Une fois le dossier "Admin" (ou autre) créé dans lequel se trouvent tous les fichiers d'administration du site, il faut empêcher une personne malveillante d'accéder au contenu et aux pages de ce répertoire.

Les fichiers .htaccess permettent de restreindre l'accès : nous pouvons très facilement créer une protection par Pseudo / Mot de passe qui empêche l'accès à tous les fichiers du dossier.

Il va falloir créer 2 fichiers :

- .htaccess : ce fichier contiendra l'adresse du .htpasswd et quelques autres options que vous pourrez définir.
- .htpasswd : ce fichier contiendra une liste de pseudo / mots de passe, pour chaque personne autorisée à accéder aux pages.

La première étape consiste en la création d'un fichier nommé : .htaccess. C'est un fichier qui ne possède pas de nom et seulement une extension, à savoir : .htaccess. Il commence donc par un point.

Editez ce nouveau fichier avec votre éditeur de texte où nous mettrons des instructions à prendre en compte par le serveur.

Instructions :

```
AuthName "Page d'administration ADMIN"
AuthType Basic
AuthUserFile "/site/www/admin/.htpasswd"
Require valid-user
```

- AuthName : texte qui invitera l'utilisateur à inscrire son pseudo / mot de passe. Ce texte peut donc être personnalisé.
- AuthUserFile : il s'agit du chemin absolu vers le fichier .htpasswd (à mettre dans le même répertoire que le fichier .htaccess).

Le chemin absolu change en fonction du serveur d'exécution, pour le trouver il est possible de faire appel à une fonction PHP : realpath.

Cette fonction indique le chemin absolu vers le fichier de votre choix.

Dans un fichier nommé **chemin.php**, inscrivons le code-source suivant :

```
<?php echo realpath('chemin.php'); ?>
```

Ensuite il faut envoyer ce fichier sur votre serveur via votre logiciel FTP en le plaçant dans le dossier que vous souhaitez protéger.

Avec votre navigateur, saisissez l'url menant à ce fichier PHP, il devrait afficher le chemin absolu. Copiez ce chemin dans votre fichier .htaccess au niveau de la ligne « AuthUserFile », après cette étape le fichier chemin.php peut être supprimé du serveur.

Dans le cas où Windows refuse la création d'un fichier commençant par un point, il est possible de le créer via l'éditeur notepad++.

htpasswd

Le fichier .htpasswd va contenir la liste des utilisateurs autorisés à accéder aux pages du dossier. On y inscrit une personne par ligne, sous cette forme :

```
pseudo:mot_de_passe_crypté
```

Voici un exemple de fichier .htpasswd

```
admin:$1$MEqT//cb$hAvid.qmmSGFW/wDlIfQ81  
julien:$1$/lgP8dYa$sQNXcCP47KhPlsneRIZoO0  
veronique:$1$1T7nqnsq$cVtoPfe0IgrjES7Ushmoy
```

Dans cet exemple, il y a 3 personnes autorisées à accéder au dossier : admin, julien et veronique.

Crypter les mots de passe

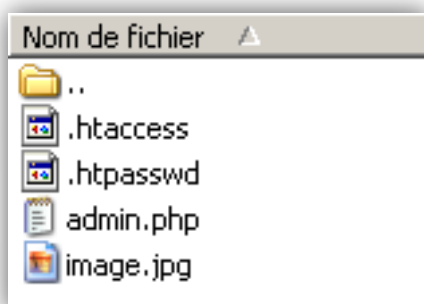
Pour crypter les mots de passe, plusieurs fonctions PHP prédéfinies existent : crypt, md5, etc.

Par exemple, pour le mot de passe "chezmoi", voici le code PHP à mettre en place afin de l'obtenir en version cryptée :

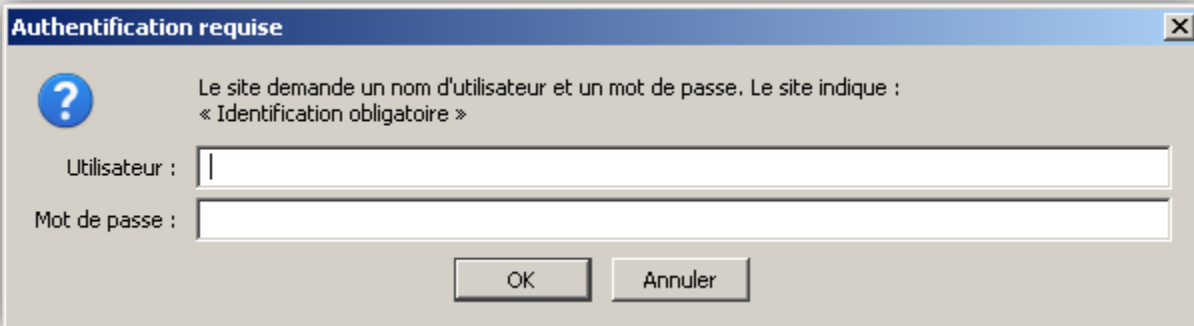
```
<?php echo crypt('chezmoi'); ?>
```

Crypter ses mots de passe est très utile : en effet, si quelqu'un vient un jour à lire votre fichier .htpasswd (quelqu'un qui utilise le même PC que vous par exemple), il ne verra que le mot de passe crypté. Aucun risque qu'il retrouve votre mot de passe : ils ne sont pas affichés en clair.

Arborescence du dossier



Si une personne tente d'accéder à une des pages du dossier (en l'occurrence admin.php), il obtiendra une fenêtre comme celle-ci lui demandant de se connecter :



Authentification requise

Le site demande un nom d'utilisateur et un mot de passe. Le site indique :
« Identification obligatoire »

Utilisateur :

Mot de passe :

OK Annuler

La protection par .htaccess a l'avantage d'être rapide et simple à mettre en place. Elle permet donc de créer une zone d'administration simple sans avoir à faire de programmation PHP ni de traitement.

Cependant, sur un site un peu plus évolué, nous aurons besoin de limiter les accès à certaines sections du site en fonction des droits d'accès ou du groupe auquel appartient un membre.