PHP PERSISTANCE PARTIELLE ET TOTALE

YAO Aristide

Table des matières

Les sessions

Les **sessions** sont un moyen simple de **stocker des données individuelles pour chaque utilisateur** en utilisant un identifiant de session unique. Elles peuvent être utilisées pour faire persister des informations entre plusieurs pages. Les identifiants de session sont normalement envoyés au navigateur via des cookies de session, et l'identifiant est utilisé pour récupérer les données existantes de la session. L'absence d'un identifiant ou d'un cookie de session indique à PHP de créer une nouvelle session, et génère ainsi un nouvel identifiant de session.. Ce système a notamment été inventé pour palier au fait que le protocole HTTP agit en mode non connecté.

Les sessions sont particulièrement utilisées pour ce type d'applications :

- Les espaces membres et accès sécurisés avec authentification.
- Gestion d'un panier sur un site de vente en ligne.
- Formulaires éclatés sur plusieurs pages.
- Stockage d'informations relatives à la navigation de l'utilisateur (thème préféré, langues...).

1. Initialisation d'une session

PHP introduit nativement une **fonction** permettant **de démarrer ou de continuer une session**. Il s'agit de **session_start()**.

Cette fonction ne prend pas de paramètre et renvoit toujours true. Elle vérifie l'état de la session courante. Si elle est inexistante, alors le serveur la crée sinon il la poursuit.

```php
<?php
  session_start();
?>
```

Remarque

- la fonction **session_start()** doit obligatoirement **être appelée avant tout envoi au navigateur** sous peine de voir afficher les fameuses erreurs.
- Il faut appeler session_start() sur chaque page utilisant le système de session.

2. La superglobale $_SESSION

Lorsqu'une session est créée, elle est par défaut vide, il faut donc lui attribuer des valeurs à sauvegarder temporairement. Pour cela, on utilise la variable superglobale $_SESSION qui peut représenter un tableau numéroté ou associatif

Écrire une variable dans une session

Pour enregistrer une nouvelle variable de session, il faut ajouter un couple clé / valeur au tableau $_SESSION.

```php
<?php
session_start();
if (!isset($_SESSION['count'])) {
  $_SESSION['count'] = 0;
} else {
  $_SESSION['count']++;
}
?>
```

Lecture d'une variable de session

Pour lire la valeur d'une variable de session, il faut appeler le tableau de session avec la clé concernée.

```php
<?php
  // Démarrage ou restauration de la session
  session_start();
  // Lecture d'une valeur du tableau de session
  echo $_SESSION['count'];
?>
```

Destruction d'une session

le serveur détruit lui même la session au bout d'un certain temps si la session n'a pas été renouvellée. En revanche, il est possible de forcer sa destruction au moyen de la fonction **session_destroy()**. Cela permet par exemple aux webmasters de proposer une page de déconnexion aux membres loggués à leur espace personnel.

```php
<?php
  // Démarrage ou restauration de la session
  session_start();
  //Destruction de la variable count
  unset($_SESSION['count']);
  // Réinitialisation du tableau de session
  // On le vide intégralement
  $_SESSION = array();
  // Destruction de la session
  session_destroy();
?>
```

3. Constantes pré-définies

- **SID** (string)
  Constante contenant le nom de la session et l'identifiant en cours, sous la forme "name=ID" ou une chaîne vide si l'identifiant de session a été défini dans un cookie de session. C'est la même valeur que celle retournée par la fonction **session_id()**.
- **PHP_SESSION_DISABLED** (int)
  Valeur retournée par session_status() si la session est désactivée.
- **PHP_SESSION_NONE** (int)

Valeur retournée par session_status() si la session est activée, mais que la session n'existe pas.

- **PHP_SESSION_ACTIVE** (int)
  Valeur retournée par session_status() si la session est activée, et que la session existe.

```
Check whether session started using Predefined Constants

if (session_status() == PHP_SESSION_NONE) {
    session_start();
}


SID constant defined dynamically!

var_dump(defined('SID'));  // bool(false) - Not defined...
session_start();
var_dump(defined('SID'));  // bool(true) - Defined now!
```

Les cookies

II

Le mécanisme des cookies a été inventé par la société Netscape dans le but de palier à certaines faiblesses du protocole HTTP mais aussi d'étendre les possibilités de relation entre le client et le site web. Leur fonction est le **stockage**, pendant une période donnée, d'une information relative à l'utilisateur (son pseudo, sa date de dernière connexion, son âge, ses préfèrences...).

En pratique, les cookies sont de simples **fichiers textes ne pouvant excéder 4Ko**. Ils sont stockés sur le disque dur de l'internaute et **gérés par les navigateurs** (Firefox, Internet Explorer, Safari, Opéra, AvantBrowser...). Leur acceptation est soumise aux filtres des navigateurs. En effet, ces derniers sont capables de refuser des cookies. Leur utilisation doit donc être scrupulesement réfléchie.

1. Créer un cookie avec setcookie()

La création d'un cookie repose sur l'envoi d'entêtes HTTP au navigateur du client au moyen de la fonction setcookie(). Il faudra l'appeler avant tout envoi de données au navigateur (print(), echo(), tag html, espace blanc...)

Syntaxe

bool setcookie (string name [, string value [, int expire [, string path [, string domain [, bool secure [, bool httponly]]]]]])

Paramètre	Explications
name	Nom du cookie
value	Valeur du cookie
expire	Date d'expiration du cookie au format timestamp UNIX, c'est à dire un nombre de secondes écoulées depuis le 01 janvier 1970
path	Chemin sur le serveur dans lequel le cookie sera valide. Si la valeur '/' est spécifiée alors le cookie sera visible sur tout le domaine. Si '/examples/' est précisé alors le cookie sera visible dans le répertoire /examples/ et tous ses sous-dossiers
domain	Domaine sur lequel le cookie sera valable. Si la valeur est '.domaine.com' alors le cookie sera disponible sur tout le domaine (sous-domaines compris). Si la valeur est 'www.domaine.com' alors le cookie ne sera valable que dans le sous-domaine 'www'
secure	Ce paramètre prend pour valeur un booléen qui définit si le cookie doit être utilisé dans un contexte de connexion sécurisée par protocole HTTPS
httponly	Ce paramètre prend pour valeur un booléen qui définit si le cookie sera accessible uniquement par le protocole HTTP. Cela signifie que le cookie ne sera pas accessible via des langages de scripts, comme Javascript. Cette configuration permet de limiter les attaques via XSS (bien qu'elle ne soit pas supportée par tous les navigateurs). Ajouté en PHP 5.2.0

```php
<?php
  // Création du cookie
  setcookie('designPrefere','prairie',time()+3600*24*31); // la date de validité est de 1 mois
?>
```

Remarque

- Lorsqu'un cookie est créé depuis une page, il ne devient disponible qu'à partir de la page suivante car il faut que le navigateur envoie le cookie au serveur.
- Un cookie dont la date d'expiration n'est pas précisée est enregistré dans la mémoire vive de l'ordinateur e non sur le disque dur. Il sera effacé à la fermeture du navigateur.

2. Sécurisez un cookie

Il est possible de sécuriser un cookie en donnant une valeur aux options **httpOnly** et **secure** sur le cookie.

cela rendra le cookie inaccessible en JavaScript sur tous les navigateurs qui supportent cette option.

Ces options permettent de réduire drastiquement les risques de faille XSS sur le site, au cas où htmlspecialchars n'aurait pas été utilisé.

```php
<?php
// retenir l'email de la personne connectée pendant 1 an
setcookie(
    'LOGGED_USER',
    'utilisateur@exemple.com',
    [
        'expires' => time() + 365*24*3600,
        'secure' => true,
        'httponly' => true,
    ]
);
```

3. Lecture d'un cookie

Lorsqu'un internaute interroge un site web repéré par un nom de domaine, son navigateur envoie au serveur la liste des cookies disponibles pour ce domaine. PHP les réceptionne puis construit un tableau associatif superglobal nommé $_COOKIE.

Les clés correspondent aux noms des cookies et les valeurs aux valeurs inscrites dans ces derniers.

```php
<?php
   // Lecture de la valeur du cookie designPrefere
   echo $_COOKIE['designPrefere'];    // Affiche 'prairie'
?>
```

Remarque

- Ajouter un couple clé / valeur au tableau $_COOKIE ne crée pas de nouveau cookie.
- Pour modifier la valeur d'un cookie, il faut réutiliser la fonction setcookie().
- Pour connaître tous les cookies utilisés, il faut lister le tableau $_COOKIE au moyen d'une boucle foreach() ou d'un appel à print_r().

```php
<?php
setcookie(
    'LOGGED_USER',
    'laurene.castor@exemple.com',
    [
        'expires' => time() + 365*24*3600,
        'secure' => true,
        'httponly' => true,
    ]
);
?>
```

4. Supprimer un cookie

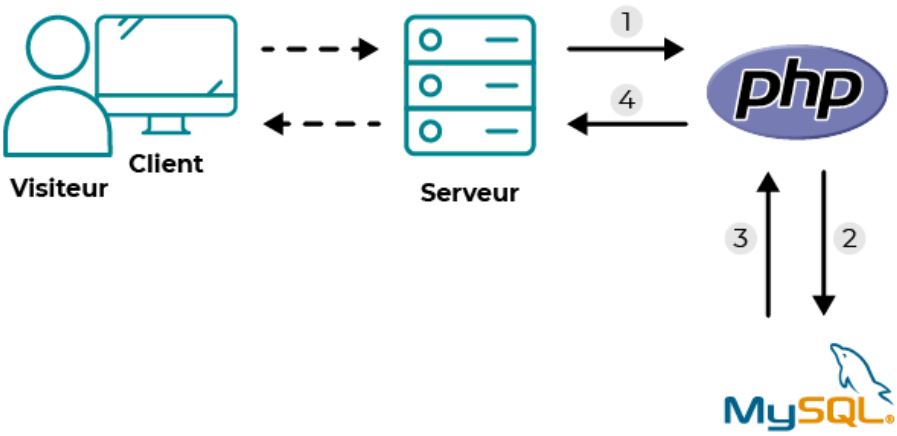
Pour supprimer un cookie, il faut de nouveau appeler la fonction setcookie() en lui passant en paramètre le nom du cookie uniquement.

```
Script de suppression complète d'un cookie
<?php
  // Suppression du cookie designPrefere
  setcookie('designPrefere');
  // Suppression de la valeur du tableau $_COOKIE
  unset($_COOKIE['designPrefere']);
?>
```

PHP et MySQL

III

1. Communication client et base de données



2. Les types de champs MySQL

Généralement, quatre types de données sont utilisés :

- INT : nombre entier ;
- VARCHAR : texte court (entre 1 et 255 caractères) ;
- TEXT : long texte (on peut y stocker un roman sans problème) ;
- DATE : date (jour, mois, année).

3. Connectez PHP à MySQL avec PDO

PDO::__construct

PDO::__construct — Crée une instance PDO représentant une connexion à une base de données

Syntaxe

```
public PDO::__construct (
   chaîne $dsn,
```

```
    ? chaîne $username =null ,
    #[\SensitiveParameter] ? chaîne $password =null ,
    ? tableau $options =null
)
```

Paramètres

- *$dsn* :
    - c'est l'adresse IP de l'ordinateur où MySQL est installé. Le plus souvent, MySQL est installé sur le même ordinateur que PHP : dans ce cas, localhost.
    - le nom de la base de données à laquelle l'on souhaite se connecter.
    - l'encodage
- *$username* :
  Nom d'utilisateur pour la chaîne DSN. Ce paramètre est facultatif pour certains pilotes PDO.
- *$password* :
  Mot de passe de la chaîne DSN. Ce paramètre est facultatif pour certains pilotes PDO.
- *$options* :
  Un tableau clé=>valeur d'options de connexion spécifiques au pilote.

Erreurs/Exceptions

Une exception PDOException est levée si la tentative de connexion à la base de données demandée échoue, quelle que soit celle PDO::ATTR_ERRMODEactuellement définie.

```php
<?php
try{
    $dsn = 'mysql:dbname=testdb;host=127.0.0.1;charset=utf8';
    $user = 'dbuser';
    $password = 'dbpass';

    $dbh = new PDO($dsn, $user, $password);
}
catch (Exception $e) {
    die("Erreur:{$e->getMessage()}");
}
?>
```

4. Effectuer une requête avec PDO

PDO::prepare — Prépare une instruction pour l'exécution et renvoie un objet d'instruction

Syntaxe

```
public PDO::prepare ( chaîne $query , tableau $options = [] ): PDOStatement | false
```

Paramètres

- *query*
  Il doit s'agir d'un modèle d'instruction SQL valide pour le serveur de base de données cible.
- *options*

Ce tableau contient une ou plusieurs paires clé=>valeur pour définir les valeurs d'attribut de l'objet PDOStatement renvoyé par cette méthode.

Valeurs de retour

Si le serveur de base de données prépare correctement l'instruction, **PDO::prepare()** renvoie un objet **PDOStatement** . Si le serveur de base de données ne parvient pas à préparer correctement l'instruction, **PDO::prepare()** renvoie **false** ou émet une exception PDOException (selon la gestion des erreurs).

```php
<?php
/* Execute a prepared statement by passing an array of values */
$sql = 'SELECT name, colour, calories
    FROM fruit
    WHERE calories < :calories AND colour = :colour';
$sth = $dbh->prepare($sql, [PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY]);
?>
```

Remarque

Les instructions préparées ne communiquent pas avec le serveur de base de données, donc PDO::prepare() ne vérifie pas la viabilité de l'instruction SQL.

5. Exécuter une requête préparée

PDOStatement::execute — exécute la requête SQL et PDOStatement::fetchAll permet de récupérer toutes les données dans un format "exploitable" c'est-à-dire sous forme d'un tableau PHP.

Lorsque la requête est paramétrée avec les marqueurs (:), PDO::execute prend en paramètre un tableau associatif.

La méthode execute retourne **true** si la requête est effectuée avec succès, **false** sinon

```php
<?php
   //requête non paramétrée
   $sth->execute();

   //requête paramétrée
   $sth->execute(['calories' => 150, 'colour' => 'red']);
?>
```

```php
<?php
/* Execute a prepared statement by passing an array of insert values */
$calories = 150;
$colour = 'red';
$sth = $dbh->prepare('SELECT name, colour, calories
    FROM fruit
    WHERE calories < ? AND colour = ?');
$sth->execute(array($calories, $colour));
?>
```

6. Récupérer les données

PDOStatement::fetchAll() renvoie un tableau contenant toutes les lignes restantes dans le jeu de résultats. Le tableau représente chaque ligne soit sous la forme d'un tableau de valeurs de colonne, soit sous la forme d'un objet avec des propriétés correspondant à chaque nom de colonne. Un tableau vide est renvoyé s'il n'y a aucun résultat à récupérer.

Syntaxe

```
public PDOStatement::fetchAll ( int $mode = PDO::FETCH_DEFAULT ): array
public PDOStatement::fetchAll ( int $mode = PDO::FETCH_COLUMN , int $column ): array
public PDOStatement::fetchAll ( int $mode = PDO::FETCH_CLASS , chaîne $class , ? tableau $constructorArgs ): array
public PDOStatement::fetchAll ( int $mode = PDO::FETCH_FUNC , appelable $callback ): array
```

Paramètres

- *mode*
  Contrôle le contenu du tableau renvoyé comme indiqué dans PDOStatement::fetch() . La valeur par défaut est PDO::ATTR_DEFAULT_FETCH_MODE (qui est par défaut PDO::FETCH_BOTH)
  Pour renvoyer un tableau composé de toutes les valeurs d'une seule colonne du jeu de résultats, spécifiez PDO::FETCH_COLUMN. Vous pouvez spécifier la colonne souhaitée avec le columnparamètre.
  Pour indexer le tableau résultant par la valeur d'une certaine colonne (au lieu de nombres consécutifs), placez le nom de cette colonne en premier dans la liste des colonnes dans SQL et utilisez PDO::FETCH_UNIQUE. Cette colonne ne doit contenir que des valeurs uniques, sinon certaines données seront perdues.
  Pour regrouper les résultats sous la forme d'un tableau tridimensionnel indexé par les valeurs d'une colonne spécifiée, placez le nom de cette colonne en premier dans la liste des colonnes dans SQL et utilisez PDO::FETCH_GROUP.
  Pour regrouper les résultats sous la forme d'un tableau à 2 dimensions, utilisez la fonction OU au niveau du bit PDO::FETCH_GROUPavec PDO::FETCH_COLUMN. Les résultats seront regroupés selon la première colonne, la valeur de l'élément du tableau étant un tableau de liste des entrées correspondantes de la deuxième colonne.

Les paramètres suivants sont des paramètres dynamiques qui dépendent du mode de récupération. Ils ne peuvent pas être utilisés avec des paramètres nommés.

- *column*
  Utilisé avec **PDO::FETCH_COLUMN**. Renvoie la colonne indexée 0 indiquée.
- *class*
  Utilisé avec **PDO::FETCH_CLASS**. Renvoie les instances de la classe spécifiée, en mappant les colonnes de chaque ligne aux propriétés nommées dans la classe.
- *constructorArgs*
  Arguments du constructeur de classe personnalisé lorsque le mode paramètre est **PDO::FETCH_CLASS**.

- *callback*
  Utilisé avec PDO::FETCH_FUNC. Renvoie les résultats de l'appel de la fonction spécifiée, en utilisant les colonnes de chaque ligne comme paramètres dans l'appel.

Valeurs de retour

PDOStatement::fetchAll() renvoie un tableau contenant toutes les lignes restantes dans le jeu de résultats.

```php
<?php
$sth = $dbh->prepare("SELECT name, colour FROM fruit");
$sth->execute();

$result = $sth->fetchAll();
print_r($result);
?>
```

```php
<?php
$sth = $dbh->prepare("SELECT name, colour FROM fruit");
$sth->execute();

/* Fetch toutes les valeurs de la 1ere colonne */
$result = $sth->fetchAll(PDO::FETCH_COLUMN, 0);
var_dump($result);
?>
```

```php
<?php
//Instanciation d'une classe pour chaque résultat
class fruit {
    public $name;
    public $colour;
}

$sth = $dbh->prepare("SELECT name, colour FROM fruit");
$sth->execute();

$result = $sth->fetchAll(PDO::FETCH_CLASS, "fruit");
var_dump($result);
?>
```

```php
<?php
//Appel d'une fonction pour chaque résultat
function fruit($name, $colour) {
    return "{$name}: {$colour}";
}

$sth = $dbh->prepare("SELECT name, colour FROM fruit");
$sth->execute();

$result = $sth->fetchAll(PDO::FETCH_FUNC, "fruit");
var_dump($result);
?>
```

7. Gérer les erreurs

Lorsqu'une requête SQL est mal écrite, PHP indiquera qu'il y a eu une erreur à la ligne du fetchAll sans préciser la cause de l'erreur.

Pour avoir une erreur plus précise, il faut activer les erreurs lors de la connexion à la base de données via PDO.

```php
<?php
$mysqlClient = new PDO(
    'mysql:host=localhost;dbname=partage_de_recettes;charset=utf8',
    'root',
    '',
    [PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION],
);
?>
```

8. PHP CRUD

Create

```
INSERT INTO recipes(title, recipe, author, is_enabled) VALUES (:title, :recipe, :author, :is_enabled)
```

Read

SELECT * FROM recipes ORDER BY title DESC

Update

UPDATE recipes SET title = :title, recipe = :recipe WHERE recipe_id = :id

Delete

DELETE FROM recipes WHERE recipe_id=:id

9. La redirection, la fonction header()

Lorsque l'on souhaite créer une redirection avec PHP, on utilise une fonction permettant d'envoyer des entêtes de type Location (adresse). Pour cela, PHP dispose de la fonction **header()** qui se charge d'envoyer les entêtes passés en paramètre.

La valeur de Location est la page sur laquelle l'internaute doit être renvoyé.

```php
<?php
  header('Location: http://www.votresite.com/pageprotegee.php');
  exit();
?>
```

🔎 *Remarque*

- l'instruction **exit()** est utilisée pour **arrêter immédiatement le reste du code PHP**. Cela garantit que la redirection s'effectue sans problème, sans que d'autres instructions perturbent ce processus.
- l'appel de cette fonction doit se faire avant tout envoi au navigateur (instruction echo, print, espace blanc, balise html...). De ce fait, il est préférable de créer une fonction de redirection.

```php
<?php
function redirectToUrl(string $url): never
```

```
{
    header("Location: {$url}");
    exit();
}
?>
```