

Conception de sites web dynamique

***HTML – CSS – JAVASCRIPT – PHP
BASE DE DONNÉES***

IF04U050

David Bouchet

david.bouchet.paris5@gmail.com

PHP ***4^e partie***



COOKIE

Les cookies sont un mécanisme d'enregistrement d'informations sur le client, et de lecture de ces informations (1).

Le cookie est l'équivalent d'un petit fichier texte stocké sur le terminal de l'internaute (2).

Ce système permet d'identifier et de suivre les visiteurs (1).

(1) <http://php.net/manual/fr/features.cookies.php>

(2) http://fr.wikipedia.org/wiki/Cookie_%28informatique%29

COOKIE

Création d'un cookie : EX :

```
<?php setcookie('pseudo', 'goldorak', time() + 365*24*3600); ?>
```

time(): nombre de secondes écoulées depuis le 1er janvier 1970

- *Setcookie()* doit être appelé avant tout code HTML (donc avant la balise `<!DOCTYPE>` tout comme *session_start()*)
- La variable globale `$_COOKIE` enregistre tous les cookies qui ont été définis

Les cookies

Écriture de cookies :

setcookie(clé, valeur, date_expiration, "chemin", "domain", secure, httponly) :

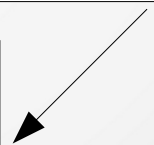
- **Date_expiration** : Date (en s) à laquelle le cookie sera effacé du client. Il s'agit d'un *Timestamp Unix* : Nombre de secondes depuis le 1er janvier 1970. On utilise en général la date du jour, définie via `time()`, à laquelle on ajoute la durée de validité désirée en secondes. Valeur 0 par défaut : le cookie expire à l'issue de la session.
- **Chemin** : répertoire du serveur à partir duquel le cookie sera accessible (la valeur par défaut est le registre courant).
- **Domain** : le nom du domaine à partir duquel peuvent être lus les cookies.
- **Secure** = FALSE si la connexion n'est pas sécurisée (par défaut), TRUE pour une connexion sécurisée de type SSL.
- **Httponly** : TRUE si accessible uniquement via le protocole http (inaccessible via JavaScript). FALSE par défaut.

COOKIE

```
<?php
    setcookie('universite', 'Paris Descartes', time()+3600);
    setcookie('pseudo', 'goldorak', time()+3600, null, null, false, true);
    setcookie('pays', 'France', time()+3600, null, null, false, true);
?>

<!DOCTYPE html >
<html> <head> <meta charset="uft-8" /> </head>
<body>
<a href="cookies2.php"> Lien vers une autre page </a>
</body></html>
```

```
<!DOCTYPE html > <html> <head> </head> <body>
Les cookies : <br/>
<?php print_r($_COOKIE); ?>
</body> </html>
```



➡ Les cookies :
Array ([pays] => France [pseudo] => goldorak [universite] => Paris Descartes)

COOKIE

Pour supprimer un cookie :

```
setcookie('universite','',time()-1);
```

ou

```
unset($_COOKIE["universite"]);
```

Inconvénient des cookies :

Le client peut paramétrer son navigateur pour les refuser

La gestion des fichiers avec PHP

Intérêt :

- Partage de fichiers
- Installation de site automatisée
- Administration de site à distance

Fonctions

- Similaires à celles du C : fopen, fread...
- Fonctions propres au PHP

Lien vers la documentation :

<http://php.net/manual/fr/ref.filesystem.php>

Fichiers : ouverture / fermeture

Vérifications préalables :

- **file_exists**(\$nom_fichier) → vérifier l'existence
- **is_file**(\$nom_fichier) → vérifier que c'est bien un fichier
- **is_readable**(\$nom_fichier) → vérifier qu'on peut le lire

Ouverture / fermeture

\$f = fopen(\$nom_fichier, \$mode) → Renvoie une ressource représentant le pointeur de fichier, ou FALSE si une erreur survient.

fclose(\$f) → renvoie FALSE si problème, TRUE sinon

Modes possibles d'ouverture :

r : lecture seule (pointeur de fichier au début du fichier)

r+ : lecture + écriture (pointeur de fichier au début du fichier)

a : écriture seule (pointeur de fichier à la fin du fichier ; crée le fichier s'il n'existe pas)

a+ : idem + lecture

w : écriture seule (remplace le contenu initial ou crée le fichier s'il n'existe pas)

w+ : idem + lecture

Fichiers : lecture / écriture (1)

Lecture de tout le contenu :

- **fpassthru(\$f)** → affiche et renvoie TRUE, ou renvoie FALSE si problème

EX :

```
if (! $f = fopen("mon_fichier.txt","r"))  
    echo "Impossible d'ouvrir le fichier";  
elseif (! fpassthru($f))  
    echo "Impossible d'afficher le fichier";
```

- **file_get_contents(\$nom_fichier)** → renvoie une chaîne avec le contenu

EX :

```
$str = file_get_contents("mon_fichier.txt")  
or exit("ERROR: file not found");  
echo $str;
```

Fichiers : lecture / écriture (2)

- **file(\$nom_fichier)**

→ renvoie un tableau dont chaque élément est une ligne du fichier

EX :

```
$tab = file("mon_fichier.txt")  
      or exit("ERROR: file not found");  
foreach ($tab as $ligne)  
    echo $ligne."<br/>";
```

Fichiers : lecture / écriture (3)

file_put_contents(\$nom_fichier, \$chaine [, int \$flags = 0])

→ remplit le fichier avec la chaîne indiquée

Revient à appeler les fonctions fopen(), fwrite() et fclose() successivement.

Si le fichier filename n'existe pas, il sera créé. Sinon, le fichier existant sera écrasé si l'option FILE_APPEND (\$flags) n'est pas définie.

```
$file = 'people.txt';  
// Ouvre un fichier pour lire un contenu existant  
$current = file_get_contents($file);  
// Ajoute une personne  
$current .= "Jean Dupond\n";  
// Écrit le résultat dans le fichier  
file_put_contents($file, $current);
```

```
$file = 'people.txt';  
// Une nouvelle personne à ajouter  
$person = "Jean Dupond\n";  
// Écrit le contenu dans le fichier, en utilisant le drapeau  
// FILE_APPEND pour rajouter à la suite du fichier et  
// LOCK_EX pour empêcher quiconque d'autre d'écrire dans le fichier  
// en même temps  
file_put_contents($file, $person, FILE_APPEND);
```

Fichiers : lecture / écriture (4)

Lecture / écriture similaire au langage C :

- **\$bool = rewind(\$f)** → place le pointeur au début du fichier
- **\$bool = fseek(\$f,\$pos)** → place le pointeur à la position \$pos
- **\$bool = feof(\$f)** → teste si fin de fichier atteinte
- **\$c = fgetc(\$f)** → lit un caractère
- **\$ligne = fread(\$f,\$L)** → Lecture du fichier en mode binaire (\$L caractères)
- **\$ligne = fgets(\$f)** → Récupère la ligne courante.
- **fwrite(\$f,\$chaine)** → écrit \$chaine dans \$f en mode binaire.
- **fputs(\$f,\$chaine)** → idem (alias de fwrite).

Fichiers : lecture / écriture (5)

EX :

```
<?php
$file = fopen("welcome.txt","r") or exit("Unable to open file!");
while(!feof($file))
    echo fgets($file)."<br />";
fclose($file);
?>
```

Fichiers : manipulation

- **Copie** : `$bool = copy($nom_source, $nom_cible)`
- **Renommage** : `$bool = rename($ancien_nom, $nouveau_nom)`
- **Destruction** : `$bool = unlink($nom_fichier)`

Gestion des répertoires

- `$bool = is_dir($nom_rep)` → teste si `$nom_rep` est un répertoire
- `$d = opendir($nom_rep)` → ouvre le répertoire courant et se place sur le premier fichier
- `$chaine = readdir($d)` → lit le nom du fichier courant et se place sur le suivant
- `rewinddir($d)` → se place sur la première entrée du répertoire
- `closedir($d)` → ferme le répertoire ouvert
- `$bool = chdir($nom_rep)` → change de répertoire
- `$bool = mkdir($nom_rep, mode)` → crée un répertoire (mode fixe les droits)
- `$bool = rmdir($nom_rep)` → supprime un répertoire

Gestion des répertoires

```
function ScanRep($Directory)
{
    echo "<b>Parcours</b>: $Directory<br>\n";
    if (is_dir($Directory) && is_readable($Directory))
    {
        if($MyDirectory = opendir($Directory))
        {
            while(($Entry = readdir($MyDirectory)) !== false)
            {
                if (is_dir($Directory."/". $Entry))
                {
                    if (($Entry != ".") && ($Entry != ".."))
                    {
                        echo "<b>Repertoire</b>: $Directory/$Entry<br>\n";
                        ScanRep($Directory."/". $Entry);
                    }
                }
                else
                {
                    echo "<b>Fichier</b>: $Directory/$Entry<br>\n";
                }
            }

            closedir($MyDirectory);
            echo "<b>Ferme</b> $Directory<br />";
        }
    }
}
```

EXEMPLE

Parcours d'un
répertoire et de ses
sous-répertoires

Interfaçage avec une base de données

PHP a des interfaces avec la plupart des SGBDR :

Oracle, Sybase, Microsoft SQL Server, PostgreSQL, MySQL

Principales fonctions PHP de manipulation des BD :

- Connexion au serveur : `<pre>_connect`
- Sélection d'une base de données : `<pre>_select_db (*)`
- Requête : `<pre>_query`
- Déconnexion : `<pre>_close`

Le préfixe change selon la base : `<pre>` = `pg` pour *PostgreSQL*

() Attention, pour PostgreSQL la sélection de la base se fait en même temps que la connexion.*

Interfaçage avec une base de données

PDO = *PHP Data Objects*

- Interface d'abstraction d'accès une base de données depuis PHP.
- Plusieurs bases de données sont prises en charge (*MySQL*, *PostgreSQL*, *Microsoft SQL Server*, *IBM DB2*, *Oracle*, etc.)
- Mêmes fonctions pour toutes les bases de données.
- Chaque base peut garder ses spécificités.
- **Disponible uniquement à partir de PHP 5.**

PostgreSQL - Connexion (1)

Vous devez vous munir des informations suivantes :

- Nom de l'hôte.
- Nom de la base de données.
- Nom d'utilisateur.
- Mot de passe.

Base de données PostgreSQL sur le serveur de Paris 5 :

- Nom de l'hôte : **opale.ens.math-info.univ-paris5.fr**
- Nom de la base de données : **BD<votre login>** (ex. **BDgdupont**)
- Nom d'utilisateur : **<votre login>** (ex. **gdupont**)
- Mot de passe : **<votre mot de passe>** (ex. **uiot5434**)

PostgreSQL - Connexion (2)

Fonction de connexion :

```
resource pg_connect($connection_string);
```

La chaîne *\$connection_string* peut contenir un ou plusieurs paramètres de configuration séparés par des espaces. Chaque paramètre de configuration est sous la forme *code = valeur*.

Les codes possibles sont les suivants :

host, port, dbname, user, password

Valeur de retour : Ressource de connexion *PostgreSQL* en cas de succès, *FALSE* en cas d'échec.

Exemple :

```
$dbconn = pg_connect("host=opale.ens.math-info.univ-paris5.fr  
dbname=BDgdupont user=gdupont password=uiot5434");  
if ($dbconn === false)  
    exit("La connexion a échoué.");
```

PostgreSQL - Connexion (3)

Il est parfois utile de déclarer ses variables de connexion dans un fichier séparé et d'inclure ce fichier dans les différentes pages de son site.

AuthenticationSettings.php

```
$host = "opale.ens.math-info.univ-paris5.fr";  
$dbname = "BDgdupont";  
$user = "gdupont";  
$password = "uiot5434";
```

page_1.php

```
require_once("AuthenticationSettings.php");  
  
$dbconn = pg_connect("host=$host dbname=$dbname user=$user  
password=$password");  
if ($dbconn === false)  
    exit("La connexion a échoué.");
```

PostgreSQL - Déconnexion (1)

Fonction de déconnexion :

```
bool pg_close([$connexion]);
```

L'argument *\$connexion* est la ressource de connexion de la base de données PostgreSQL (obtenue à l'aide de *pg_connect()*).

Lorsque *\$connexion* n'est pas présent, c'est la dernière connexion faite par *pg_connect()* qui est utilisé.

Il n'est généralement pas nécessaire de fermer une connexion, car elles sont automatiquement fermées à la fin d'un script.

Valeur de retour : *TRUE* en cas de succès, *FALSE* en cas d'erreur.

PostgreSQL - Déconnexion (2)

Exemple :

```
// Connexion à la base.
$dbconn = pg_connect("host=opale.ens.math-info.univ-paris5.fr
dbname=BDgdupont user=gdupont password=uiot5434");
if ($dbconn === false)
    exit("La connexion a échoué.");

// ...
// Traitement sur la base de données.
// ...

// Termine la connexion.
pg_close($dbconn);
```


PostgreSQL – Exécuter une requête

Fonction :

```
resource pg_query([$connexion], $query);
```

\$connexion : Ressource de connexion de la base de données *PostgreSQL* (obtenue à l'aide de *pg_connect()*). Lorsque *\$connection* n'est pas présent, c'est la dernière connexion faite par *pg_connect()* qui est utilisé.

\$query : Requête SQL à exécuter.

Valeur de retour : Une ressource de résultats en cas de succès ou *FALSE* si une erreur survient.

PostgreSQL – Erreur de requête

Il est possible de récupérer un message d'erreur si la connexion n'a pas pu être établie.

Il faut utiliser la fonction `pg_last_error()`.

Exemple :

```
$result = pg_query($query);  
  
if ($result === false)  
    exit("Erreur de requête : ".pg_last_error());
```

Remarque :

Sur la plupart des serveurs, les erreurs apparaissent déjà sous la forme d'un avertissement (*warning*). Il devient donc inutile de gérer l'affichage des erreurs.

PostgreSQL – Création d'une table



Instruction SQL : **CREATE TABLE**

```
$query = "CREATE TABLE customer
        (id SERIAL PRIMARY KEY,
         name VARCHAR(50),
         age SMALLINT,
         email VARCHAR(64) UNIQUE);"
```

```
$result = pg_query($query);
```

```
if ($result === false)
    echo pg_last_error();
```

```
else
    echo "La table 'customer' a été créée.";
```

Column	Type	Not Null	Default	Constraints
id	integer	NOT NULL	nextval('customer_id_seq'::regclass)	
name	character varying(50)			
age	smallint			
email	character varying(64)			

PostgreSQL – Remplir une table

Instruction SQL : **INSERT INTO**

```
$query[] = "INSERT INTO customer VALUES(DEFAULT, 'roger', 24, 'roger@gm.com')";  
$query[] = "INSERT INTO customer VALUES(DEFAULT, 'david', 35, 'david@yh.com')";  
$query[] = "INSERT INTO customer VALUES(DEFAULT, 'max', 24, 'max@gm.com')";  
$query[] = "INSERT INTO customer VALUES(DEFAULT, 'roger', 15, 'roger.b@ht.fr')";  
$query[] = "INSERT INTO customer VALUES(DEFAULT, 'jade', 32, 'jade15@ht.fr')";  
$query[] = "INSERT INTO customer VALUES(DEFAULT, 'léa', 24, 'lea@gm.com')";  
$query[] = "INSERT INTO customer VALUES(DEFAULT, 'jade', 27, 'jade.s@gm.com')";
```

```
foreach ($query as $q)  
    pg_query($q);
```

id	name	age	email
1	roger	24	roger@gm.com
2	david	35	david@yh.com
3	max	24	max@gm.com
4	roger	15	roger.b@ht.fr
5	jade	32	jade15@ht.fr
6	léa	24	lea@gm.com
7	jade	27	jade.s@gm.com