



PHP – la suite

Fonctions et interactions avec la
base de données



CONSULTING **B**STORM

Table des matières

Les instructions de contrôle	2
break.....	2
continue	2
L'opérateur ternaire	2
Les fonctions de PHP	3
Les fonctions connues de PHP	3
include()	3
var_dump()	3
print_r()	3
isset()	3
date()	4
La documentation.....	4
Exercices	5
Consigne	5
Exercice 1	6
Exercice 2	6
Exercice 3	6
Variables de session.....	6
Interaction entre PHP et la Base de données.....	7
Se connecter, sélectionner et afficher.....	7
Pourquoi PDO ?	7
Se connecter à la base de données	7
Sélectionner des données dans la base de données	8
Afficher les données récoltées	9
La boucle foreach.....	9
Les tableaux associatifs	10
La boucle foreach clé valeur	10
Modifier les données dans la base de données.....	11
Récupérer les données sous forme d'un tableau	11
Préparer pour éviter les injections SQL	12
Exercices	12
Consigne	12
Projet 1 : Le tableau.....	13
Projet 2 : Le calendrier ordonné.....	13
Projet 3 : Le calendrier jour.....	14
Projet 4 : Les températures.....	14

Les instructions de contrôle

break

Il permet de stopper une action en cours. Nous l'avons déjà vu dans le « switch ». Vous pouvez le placer dans les boucles (for, while, ...) pour mettre fin à l'opération.

```
$tableau = array(1, 2, 3, 4, 5) ;
$i = 0 ;
while ($i < 5)
{
    if ($tableau[$i] == 3)
    {
        break ;
    }
    echo "$tableau[$i]<br>" ;
    $i++ ;
}
```

```
1
2
```

continue

Il permet d'annuler l'itération et de passer à la suivante.

```
$tableau = array(1, 2, 3, 4, 5) ;

for ($i = 0 ; $i < 5 ; $i++)
{
    if ($tableau[$i] == 3)
    {
        continue ;
    }
    echo "$tableau[$i]<br>" ;
}
```

```
1
2
4
5
```

L'opérateur ternaire

Il est possible d'écrire une condition en une ligne grâce à une ternaire. Elle se compose comme suit :

```
// (condition) ? instructionTrue : instructionFalse
```

Exemple :

```
$action = (empty($_POST['action'])) ? 'défaut' : $_POST['action'];
```

La ligne ci-dessus est identique à la condition suivante :

```
if (empty($_POST['action']))
{
    $action = 'défaut';
}
else
{
    $action = $_POST['action'];
}
```

Les fonctions de PHP

Qu'est-ce qu'une fonction ? Ce sont de petits programmes que :

- Soit PHP connaît déjà,
- Soit que vous pouvez créer car vous souhaitez les réutiliser.

Les fonctions connues de PHP

include()

La fonction include sert à **inclure** une partie de code du script où l'on souhaite qu'il se trouve.

index.php

```
< ?php
include("header.php") ;
include("footer.php") ;
?>
```

header.php

```
<!DOCTYPE html>
<html>
<head>
    <title></title>
</head>
<body>
```

footer.php

```
</body>
</html>
```

var_dump()

La fonction var_dump affiche le **contenu** et le **type** d'une variable.

Le paramètre passé à la fonction var_dump est la variable à analyser.

```
var_dump($maVariable) ;
```

```
Int(3)
array(1) { ["maison"]=> string(1) "3" }
```

print_r()

La fonction affiche le **contenu** d'une variable.

Le paramètre passé à la fonction print_r est la variable à analyser.

```
print_r($maVariable) ;
```

```
3
Array ( [maison] => 3 )
```

isset()

La fonction isset vérifie qu'une variable existe et que son contenu est différent de null.

Elle s'utilise comme suit isset(\$nomVariable) et retourne une valeur logique (true ou false).

```
if(isset($nomVariable))
{
    // instructions
}

//exemple
if(isset($_POST['nomVariable']))
{
    // instructions
}
```

date()

La fonction date permet de générer la date du jour en fonction de ses paramètres.

Caractères pour le paramètre format	Description	Exemple de valeurs retournées
<i>Jour</i>	---	---
<i>d</i>	Jour du mois, sur deux chiffres (avec un zéro initial)	01 à 31
<i>D</i>	Jour de la semaine, en trois lettres (et en anglais - par défaut : en anglais, ou sinon, dans la langue locale du serveur)	Mon à Sun
<i>j</i>	Jour du mois sans les zéros initiaux	1 à 31
<i>l</i> ('L' minuscule)	Jour de la semaine, textuel, version longue, en anglais	Sunday à Saturday
<i>N</i>	Représentation numérique ISO-8601 du jour de la semaine (ajouté en PHP 5.1.0)	1 (pour Lundi) à 7 (pour Dimanche)
<i>S</i>	Suffixe ordinal d'un nombre pour le jour du mois, en anglais, sur deux lettres	st, nd, rd ou th. Fonctionne bien avec <i>j</i>
<i>w</i>	Jour de la semaine au format numérique	0 (pour dimanche) à 6 (pour samedi)
<i>z</i>	Jour de l'année	0 à 365
<i>Semaine</i>	---	---
<i>W</i>	Numéro de semaine dans l'année ISO-8601, les semaines commencent le lundi (ajouté en PHP 4.1.0)	Exemple : 42 (la 42ème semaine de l'année)
<i>Mois</i>	---	---
<i>m</i>	Mois au format numérique, avec zéros initiaux	01 à 12
<i>n</i>	Mois sans les zéros initiaux	1 à 12
<i>t</i>	Nombre de jours dans le mois	28 à 31
<i>Année</i>	---	---
<i>L</i>	Est ce que l'année est bissextile	1 si bissextile, 0 sinon.
<i>Y</i>	Année sur 4 chiffres	Exemples : 1999 ou 2003

Et encore beaucoup d'autres paramètres possibles...

La documentation

Il est venu le temps de voler de vos propres ailes. Il existe pour chaque langage une documentation sur celui-ci. Pour PHP, vous pouvez trouver la documentation officielle sur :

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



Le plus facile est de rechercher dans Google « le nom de la fonction » suivi de « php ». Le premier lien sera celui de la documentation.

Exercices

Consigne

Recherchez dans la documentation la fonction qui :

1	fournit la taille (en caractères) d'une chaîne de caractères	
2	met en minuscules tous les caractères d'une chaîne	
3	met en majuscules tous les caractères d'une chaîne	
4	met en majuscule le premier caractère d'une chaîne	
5	permet d'extraire une partie d'une chaîne de caractères	
6	fournit le nombre de fois qu'une sous-chaîne apparaît dans une chaîne de caractères	
7	extraie d'une chaîne la suite des caractères rencontrée à partir de la dernière apparition d'un repère fourni en paramètre. Le repère utilisé consiste en un caractère	
8	supprime toutes les balises HTML et PHP présentes dans une chaîne de caractères	
9	supprime les anti-slash (\) dans une chaîne de caractères	
10	convertit les caractères de nouvelle ligne en retour de chariot HTML 	
11	encode une chaîne de caractères en convertissant les caractères non-ascii en caractère de type %xx pour générer une URL	
12	décode une chaîne URL. Il convertit du type %xx vers le caractère correspondant.	
13	convertit les caractères spéciaux en leur équivalent HTML	
14	convertit tous les caractères ayant une signification particulière dans le langage HTML	
15	remplace toutes les occurrences d'une chaîne par une autre chaîne.	
16	retourne une copie d'une chaîne de caractères sans les espaces « blancs » de début et de fin de chaîne.	
17	scinde une chaîne de caractères sur base d'un délimiteur et retourne un tableau contenant les éléments résultant de la division	
18	retourne une chaîne de caractères constituée de tous les éléments d'un tableau séparés par une chaîne de jointure.	
19	permet d'assigner une série de variables en une seule ligne à partir d'un tableau	
20	retourne un tableau de 4 éléments(0, 1, key, value) reprenant la clé et la valeur de l'élément courant d'une table.	
21	retourne le nombre d'éléments que contient le tableau	
22	retourne « true » si le type de la variable est un tableau, le cas échéant « false ».	

function

Une fonction possède un nom, elle peut avoir des paramètres. Elle doit être déclarée avant d'être utilisée.

```
function nom_de_la_fonction([parametre])
{
    //instructions
}
```

Exemple :

```
function addition($nombre1, $nombre2)
{
    $resultat = $nombre1 + $nombre2 ;
    return $resultat ;
}
```

Une fonction peut recevoir ou pas des paramètres et renvoyer une valeur à l'aide du return ou pas.

Vous pouvez récupérer la valeur et la placer dans une variable.

```
$resultat = addition(2, 4) ;
```

Exercice 1

Consigne

Réalisez une fonction calculant le carré d'un nombre entier donné en paramètre.

Exercice 2

Consigne

Réalisez une fonction de recherche dans un tableau. Cette fonction va recevoir un tableau, la taille du tableau et la valeur recherchée en paramètres et renvoyer l'indice de l'élément dans le tableau. Si l'élément ne s'y trouve pas, la fonction renvoie -1.

Exercice 3

Consigne

Améliorez les projets 1, 2, 3 et 4 en intégrant des fonctions.

Variables de session

Les variables de session permettent de garder en mémoire une variable.

```
$_SESSION['nom'] = "valeur" ;
```

Pour que ça marche, il faut commencer une session :

```
session_start()
```

Il démarre une nouvelle session ou reprend une session existante.

Pour stopper une session :

```
session_destroy()
```

Il détruit une session.

Enregistrer une variable :

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

Retirer une variable de session avec la superglobale

```
unset($_SESSION['count']);
```

Interaction entre PHP et la Base de données

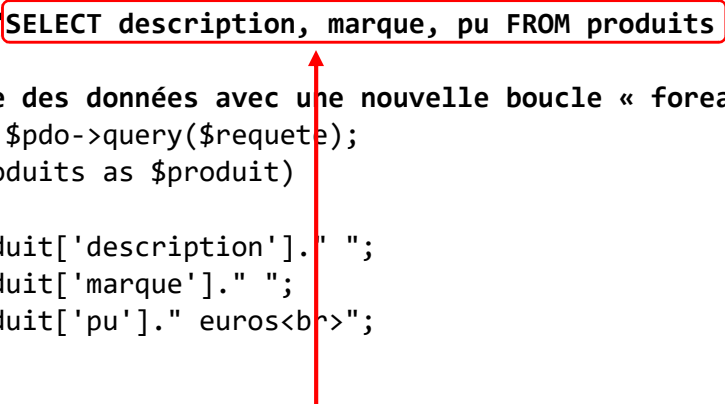
Se connecter, sélectionner et afficher

Observez ce code :

```
<?php
// connexion avec la base de données avec PDO
$basededonnees = "mysql:host=localhost;dbname=nom_bd";
$utilisateur = "login";
$motdepasse = "password";
$pdo = new PDO($basededonnees, $utilisateur, $motdepasse);

// sélection des données
$requete = "SELECT description, marque, pu FROM produits ";

// affichage des données avec une nouvelle boucle « foreach »
$produits = $pdo->query($requete);
foreach($produits as $produit)
{
    echo $produit['description']. " ";
    echo $produit['marque']. " ";
    echo $produit['pu']. " euros<br>";
}
?>
```



Il est donc possible de faire une requête SQL avec PHP.

Pourquoi PDO ?

PDO est une couche d'abstraction d'accès à la base de données. Il supporte plusieurs types de bases de données. C'est un avantage si l'on veut réutiliser son code sur un projet utilisant un autre SGBD. PDO signifie PHP Data Objects.

L'accès à la base de données se réalisera en 4 étapes :

- Connexion
- Requête SQL
- Traitement du résultat
- Déconnexion

PDO va procéder en 3 étapes :

- Créer le constructeur « PDO » afin d'établir la connexion,
- Utiliser la méthode « query » qui exécutera la requête SQL
- Utiliser la méthode « fetch » qui récupérera une ligne de résultat.

Se connecter à la base de données

Pour se connecter, on crée un objet PDO de la manière suivante :

```
$pdo = new PDO($basededonnees, $utilisateur, $motdepasse);
```

La source de données est généralement initialisée de la manière suivante :

```
$basededonnees = "mysql:host=localhost;dbname=nom_bd";
```


(valable parce qu'on a choisi de travailler avec MySQL).

La variable contenant la source de données doit contenir :

- le nom d'hôte / du serveur (souvent localhost)
- le nom de la base de données

Ces deux informations sont séparées par un point-virgule.

Pour le moment :

L'utilisateur : \$utilisateur sera pour le moment « root ».

Le mot de passe : \$motdepasse est vide.

Nous utiliserons donc :

```
// connexion avec la base de données avec PDO
$basededonnees = "mysql:host=localhost;dbname=nom_bd";
$identifiant = "root";
$motdepasse = "";
$pdo = new PDO($basededonnees, $identifiant, $motdepasse);
```

Pour faciliter la récupération des erreurs, nous allons placer le code entre les instructions « try » et « catch » :

- « try » signifie qu'il va essayer d'exécuter le code à l'intérieur des accolades.
- « catch » signifie que s'il n'y arrive pas, il va récupérer les erreurs.

```
try
{
    $pdo = new PDO($basededonnees, $identifiant, $motdepasse);
}
catch(PDOException $e)
{
    echo $e->getMessage();
}
```

Sélectionner des données dans la base de données

Pour envoyer une requête à la base de données, on utilise la fonction query de la façon suivante :

```
$resultats = $pdo->query($requete) ;
```


Il récupère un objet de données, sous la forme d'un « tableau ».

On peut donc accéder à chaque colonne du tableau en mentionnant le nom de ce que vous aurez mis après le SELECT dans votre requête.

Afficher les données récoltées

Pour lire un champ parmi les résultats reçus, on utilise les propriétés des tableaux associatifs. Chaque champ est indicé par son nom. Par exemple :

```
$unElement["description"] ;
```



description	prix
bic	1
souris	5
clavier	10

La boucle foreach

Il existe une autre boucle qui permet d'afficher les éléments d'un tableau.

```
// On récupère les données sous forme d'un tableau.
$tableau = $pdo->query($requete);

// On affiche chaque ligne du tableau comme suit :
foreach($tableau as $ligne)
{
    echo $ligne['nom1']." ";
    echo $ligne['nom2']." ";
    echo $ligne['nom3']." euros<br>";
}
```

Exemple :

```
// On récupère les données sous forme d'un tableau.
$produits = $pdo->query($requete);

// On affiche chaque ligne du tableau comme suit
foreach($produits as $produit)
{
    echo $produit['description']." ";
    echo $produit['marque']." ";
    echo $produit['pu']." euros<br>";
}
```

Les données récoltées lors de la requête :

description	marque	pu
crayon	Bic	2
gomme	Plume	1
stylo bille	Bic	3

En résumé :

```
foreach($tableau as $ligne)
{
    echo $ligne['nom_col1']." ";
    echo $ligne['nom_col2']." ";
    echo $ligne['nom_col3']."<br>";
}
```

Les données récoltées lors de la requête :

nom_col1	nom_col2	nom_col3

Les tableaux associatifs

Chaque élément n'est plus indicé par un numéro mais par une chaîne de caractères.

<i>planete1</i>	<i>planete2</i>	<i>planete3</i>
Mercure	Vénus	Terre

Exemple :

```
$tableau = array(
    "planete1" => "Mercure",
    "planete2" => "Venus",
    "planete3" => "Terre"
);
```

La boucle foreach clé valeur

La boucle foreach clé valeur affiche les éléments « indice » chaîne de caractères et sa valeur.

Elle ne fonctionne que pour les tableaux et les objets, et émettra une erreur si vous tentez de l'utiliser sur une variable de type différent ou une variable non initialisée. Il existe deux syntaxes :

Première syntaxe :

```
foreach ($tableau as $value)
{
    //instructions
}
```

Exemple :

```
$tableau = array(1, 2, 3, 4);
foreach ($tableau as $valeur)
{
    echo " Valeur : $valeur<br>";
}
```

1
2
3
4

Deuxième syntaxe :

```
foreach ($tableau as $cle => $valeur)
{
    //instructions
}
```

Exemple :

```
$tableau = array(
    "un" => 1,
    "deux" => 2,
    "trois" => 3,
    "dix-sept" => 17
);
foreach ($tableau as $cle => $valeur)
{
    echo "Clé : $cle Valeur : $valeur<br>";
}
```

Clé : un Valeur : 1
Clé : deux Valeur : 2
Clé : trois Valeur : 3
Clé : dix-sept Valeur : 17

Modifier les données dans la base de données

Il y a 3 actions possibles :

INSERT	insérer des données	INSERT INTO <i>nom_table</i> (<i>colonne1</i> , <i>colonne2</i> , <i>colonne3</i> , ...) VALUES (<i>valeur1</i> , <i>valeur2</i> , <i>valeur3</i> , ...);
UPDATE	mettre à jour des données	UPDATE <i>nom_table</i> SET <i>colonne1</i> = <i>valeur1</i> , <i>colonne2</i> = <i>valeur2</i> , ... WHERE <i>condition</i> ;
DELETE	supprimer des données	DELETE FROM <i>nom_table</i> WHERE <i>condition</i> ;

Pour cela, nous utilisons la méthode « exec » qui lance l'exécution de la requête et retourne le nombre de lignes modifiées par cette requête.

```
try
{
    $pdo = new PDO($basededonnees, $identifiant, $motdepasse);
    $requete = "DELETE * FROM client";

    $clients = $pdo->exec($requete);
}
catch(PDOException $e)
{
    echo $e->getMessage();
}
```

Nous passerons toujours par le « try » et « catch » pour récupérer les erreurs.

Récupérer les données sous forme d'un tableau

Nous connaissons déjà la méthode « query » qui récupère un objet. Il existe une autre méthode « fetch » qui renvoie un tableau. Il est possible de choisir le format de retour du tableau.


PDO ::FETCH_BOTH	renvoie un tableau indexé par les noms de colonnes mais aussi des numéros	<table> <tr> <td>0</td><td>1</td><td>2</td></tr> <tr> <td>nom1</td><td>nom2</td><td>nom3</td></tr> <tr> <td></td><td></td><td></td></tr> </table>	0	1	2	nom1	nom2	nom3			
0	1	2									
nom1	nom2	nom3									
PDO ::FETCH_ASSOC	renvoie un tableau indexé par les noms des colonnes.	<table> <tr> <td>nom1</td><td>nom2</td><td>nom3</td></tr> <tr> <td></td><td></td><td></td></tr> </table>	nom1	nom2	nom3						
nom1	nom2	nom3									
PDO ::FETCH_NUM	renvoie un tableau indexé par des numéros	<table> <tr> <td>0</td><td>1</td><td>2</td></tr> <tr> <td></td><td></td><td></td></tr> </table>	0	1	2						
0	1	2									

NB : Il existe d'autres manières de générer ce tableau. (voir documentation)

Exemple :

```
try
{
    $pdo = new PDO($basededonnees, $identifiant, $motdepasse);
    $requete = "SELECT * FROM client";

    $tableau = $pdo->query($requete);
    while ($ligne = $tableau->fetch(PDO::FETCH_BOTH))
    {
        echo $ligne[0]."-".$ligne["nom2"]."<br>";
    }
}
catch(PDOException $e)
{
    echo $e->getMessage();
}
```



0	1	2
nom1	nom2	nom3

Préparer pour éviter les injections SQL

Les avantages de la méthode « prepare » :

- Optimise les requêtes qui seront exécutées plusieurs fois,
- Préviend des injections SQL.

La méthode « prepare » prépare requête à l'exécution et retourne un objet.

```
$requete = "INSERT INTO client (nom, prenom) VALUES (:nom, :prenom)";

$insertion = $pdo->prepare($requete);
$insertion->execute(array(
    ":nom" => $nomRecupere,
    ":prenom" => $prenomRecupere
));
```

Ces variables pourraient être récupérées depuis un formulaire. Un utilisateur ayant de mauvaises intentions aurait pu écrire une requête qui vide la BD. Grâce à « prepare » les variables ne correspondant pas au type attendu, ces variables seront modifiées et n'apporteront pas atteintes à la BD.

La méthode « execute » retourne « true » si l'opération s'est bien exécutée ou « false » en cas d'échec.

Exercices

Consigne

Adaptez les projets suivants en passant par une base de données.

Créez cette base de données qui contiendra toutes les tables de ce chapitre.

Projet 1 : Le tableau

- 1) Créez une table « tableau » avec 2 colonnes :
 - id INT Auto-Incrémenté,
 - valeur INT.
- 2) Insérez les données suivantes : 1, 4, 3, 5, 2
- 3) Récupérez les valeurs depuis la base de données et stockez-les dans un tableau.
- 4) Affichez :

1	4	3	5	2
---	---	---	---	---

Plus Grand
 Plus Petit
 Somme
 Moyenne

Rechercher

Projet 2 : Le calendrier ordonné

- 1) Créez une table « mois_annee » avec 2 colonnes :
 - id INT Auto-Incrémenté,
 - nom_mois VARCHAR 20.
- 2) Insérez les données suivantes : janvier, février, mars, avril, mai, juin, juillet, aout, septembre, octobre, novembre, décembre
- 3) Récupérez les valeurs depuis la base de données et stockez-les dans un tableau.
- 4) Affichez :

Le calendrier

Dans quel ordre souhaitez-vous qu'il s'affiche ?

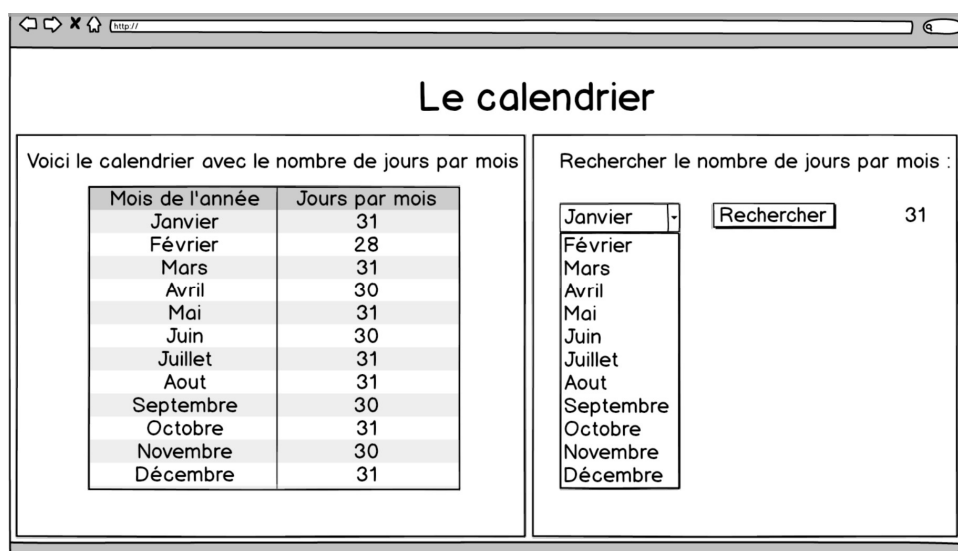
☐ chronologique
 ☐ inverse
 Valider

Mois de l'année
Janvier
Février
Mars
Avril
Mai
Jun
Juillet
Aout
Septembre
Octobre
Novembre
Décembre

C'est un tableau
 <-

Projet 3 : Le calendrier jour

- 1) Créez une table « jours_mois » avec 2 colonnes :
 - id INT Auto-Incrémenté,
 - nombre INT.
- 2) Insérez les données suivantes : 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31
- 3) Récupérez les valeurs depuis la base de données et stockez-les dans un tableau.
- 4) Affichez :



Projet 4 : Les températures

- 1) Créez une table « septembre_temp » avec 2 colonnes :
 - id INT Auto-Incrémenté,
 - temperature INT.
- 2) Insérez les données suivantes : 20, 20, 20, 20, 24, 19, 18, 17, 16, 17, 17, 18, 17, 17, 14, 15, 16, 16, 16, 17, 17, 19, 17, 20, 19, 20, 21, 21, 24, 17
- 3) Récupérez les valeurs depuis la base de données et stockez-les dans un tableau.

Affichez :

