

Chapitre III : PHP Hypertext Preprocessor

1

Pré-requis : Bases de données, Programmation orientée objet

Objectifs : Créer des sites dynamiques avec PHP, JavaScript et MySQL

Plan

- Introduction au langage PHP
- Traitement des données des formulaires en PHP
- Accès aux bases de données en PHP
- Sessions & Cookies en PHP
- PHP Orienté Objets

Les avantages de PHP dans le développement moderne

- Langage open source avec une large communauté
- Compatible avec la majorité des serveurs web et bases de données
- Écosystème riche (frameworks, bibliothèques, outils)
- Parfait pour les API et les applications web modernes
- Documentation abondante et apprentissage accessible

Frameworks PHP Populaires et Cas d'Utilisation

- - Laravel : Idéal pour les applications web complexes et APIs
- - Symfony : Pour des applications robustes et modulaires
- - CodeIgniter : Simple et léger pour les petits projets
- - Yii : Rapide, sécurisé et adapté aux grandes applications
- - CakePHP : Développement rapide avec des conventions prédéfinies

Introduction au langage PHP

5

XAMPP

- Suite logicielle complète qui inclut effectivement les quatre éléments suivants :
 1. **Un serveur Web Apache** : Pour exécuter des sites web localement.
 2. **Un serveur de bases de données MySQL/MariaDB** : Pour gérer les bases de données relationnelles.
 3. **Un interpréteur de script PHP** : Pour exécuter des scripts côté serveur.
 4. **Une interface d'administration SQL phpMyAdmin** : Pour gérer les bases de données MySQL via une interface web.

Introduction au langage PHP

6

PHP peut être utilisé de deux manières différentes :

- soit comme un langage de programmation traditionnel (en ligne de commande)
- soit intégré dans une page web

Exemple de programme
PHP

Fichier
bonjour.php

```
<?php  
  
echo 'Bonjour';  
  
?>
```

Intégration PHP et HTML

7

Exemple de script, code source (côté serveur) :

```
<html>
<body>
<h1>Mon premier script</h1>
<?php echo "Bonjour\n"; ?>
</body>
</html>
```

Autre écriture du même script :

```
<?php
echo "<html>\n<body>\n";
echo "<h1>Mon premier script</h1>\n";
echo "Bonjour\n";
echo "</body>\n</html>\n";
?>
```

Intégration PHP et HTML

8

Génération du code HTML par PHP

- `echo Expression;`
 - `echo "Chaine de caracteres";`
 - `echo (3+2)*7;`
- `print(expression);`
 - `print("Chaine de caracteres");`
 - `print ((3+2)*7);`
- `printf (chaîne formatée);`
 - `printf ("Le périmètre du cercle est %d", $Perimetre);`

Introduction au langage PHP

9

Syntaxe de base:

Quelque règles

- Toute instruction se termine par un point-virgule
- Sensible à la casse
 - Sauf par rapport aux fonctions

Commentaires

- `/* Voici un commentaire! */`
- `// un commentaire sur une ligne`

Nom variable: precede par \$

- \$aPas de déclaration
- `$x = 1; echo "$x";`

Constantes

- `Define("nom_constant", valeur_constant)`
 - `define ("ma_const", "PHP5") ;`
 - `define ("an", 2015) ;`

Pour vérifier si une constante est défini:

- `if (defined(nom_cte))`

Introduction au langage PHP

10

- Les variables sont toutes préfixées par le symbole \$

```
<?php
```

```
$annee = 1995;  
$nom = 'Ameni';
```

↙ : retour à la ligne

```
echo 'Bonjour, je m'appelle'.  
$nom.'<br>';  
echo "je suis né en ".$annee;
```

```
?>
```

- Les variables ne sont pas typées en PHP

```
<?php  
$valeur = 1;  
$valeur = 'Coucou';  
?>
```

Transtypage : Une variable peut prendre n'importe quelle valeur (entier, réel, chaîne, tableau)

Introduction au langage PHP

11

Il est possible de convertir une variable en un type primitif grâce au `cast`⁽¹⁾ (comme en C).

Exemple :

```
$str = "12";           // $str vaut la chaîne "12"  
$nbr = (int)$str;      // $nbr vaut le nombre 12
```

(1) : Le cast est une conversion de type. L'action de caster consiste en convertir une variable d'un type à un autre.

Quelques fonctions :

empty(\$var) : renvoie vrai si la variable est vide

isset(\$var) : renvoie vrai si la variable existe

unset(\$var) : détruit une variable

gettype(\$var) : retourne le type de la variable

settype(\$var, "type") : convertit la variable en type **type** (cast)

is_long(), **is_double()**, **is_string()**, **is_array()**, **is_object()**, **is_bool()**,
is_float(), **is_numeric()**, **is_integer()**, **is_int()**...

Introduction au langage PHP

12

Les instructions de base

Le test conditionnel : if (condition) { ... } else { ... }

```
if ($valeur == 1) {  
    echo 'la valeur est égale à 1';  
} else {  
    echo 'la valeur est différente de  
1';  
}
```

```
if ($valeur > 1) {  
    echo 'la valeur est supérieure à  
1';  
}  
if ($valeur < 1) {  
    echo 'la valeur est inférieure à 1';  
}  
if ($valeur == 1) {  
    echo 'la valeur est égale à 1';  
}  
if ($valeur != 1) {  
    echo 'la valeur est différente de  
1';  
}
```

Introduction au langage PHP

13

La boucle tant que : while (condition) { instructions }

```
$i = 1;  
  
$somme = 0;  
  
while ($i <= 10) {  
    $somme = $somme + $i;  
    $i = $i + 1;  
}  
  
echo 'Somme des entiers de 1 à 10 = '.$somme  
;
```

Introduction au langage PHP

14

La boucle pour : for (initialisation ; condition d'arrêt; incrémentation) { .. }

```
$somme = 0;

for ($i = 1; $i <= 10; $i = $i + 1)
{
    $somme = $somme + $i;
}

echo 'Somme des entiers de 1 à 10 = '.$somme
;
```

Introduction au langage PHP

15

Il existe plusieurs manières de manipuler les tableaux

```
$tableau =  
array();  
$tableau[1] = 1;  
$tableau[2] = 2;  
$tableau[3] = 3;  
print_r( $tableau  
);
```

```
$tableau = array(1, 2, 3);  
  
foreach($tableau as $element) {  
  
    echo  
  
    'T['.$element.']='.$element;  
  
}
```

```
$tableau = array();  
  
array_push($tableau,1);  
  
array_push($tableau,2);  
  
array_push($tableau,3);  
  
reset( $tableau );  
  
foreach ($tableau as $cle => $val) {  
  
    echo 'tableau[' . $cle . ']' = ' . $val;  
  
    echo '<br>'; }  

```

Introduction au langage PHP

16

Les éléments du tableaux ne sont pas forcément d'indices consécutifs :

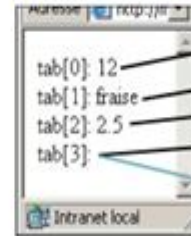
```
$tab4[0] = 12 ;  
$tab4[1] = "fraise" ;  
$tab4[2] = 2.5 ;  
$tab4[5] = "e15" ;
```

Clé	Valeur
0	12
1	"fraise"
2	2.5
3	
4	
5	"e15"

Alors comment parcourir de tels tableaux ?

Parcours classique!

```
for ($i=0; $i < sizeof($tab4); $i++)  
{ echo "tab4[$i]: "  
  . $"  
}
```



Clé	Valeur
0	12
1	"fraise"
2	2.5
3	
4	
5	"e15"

Solution = Parcours adapté avec foreach

Introduction au langage PHP

17

Structure de contrôle Pour chaque:

```
foreach ($tab as $cle => $val) {  
    echo "tab[$cle] = $val <br>";  
}
```

```
...  
$tab4[0] = 12 ;  
$tab4[1] = "fraise" ;  
$tab4[2] = 2.5 ;  
$tab4[5] = "el5" ;  
foreach($tab4 as  
$v)  
{  
    echo "Val:  
$v<br>\n";  
}  
...
```

```
...  
Val:12<br>  
Val:fraise<br>  
>  
Val:2.5<br>  
Val:el5<br>  
...
```

Navigateur

```
Val: 12  
Val: fraise  
Val: 2.5  
Val: el5
```

Intranet local

Introduction au langage PHP

18

Syntaxe de base: Les Tableaux Associatifs

Tableaux dont l'accès aux éléments n'est pas réalisé par les indices (0,1,...) mais plutôt à l'aide d'une clé de type entier ou chaîne.

Exemples:

```
$tab['un'] = 12 ;  
$tab[205] = "bonjour" ;  
$tab["la valeur"] = 3.0 ;
```

Creation

```
$tab = array(cle1 => val1,  
            cle2 => val2,  
            ...);
```

```
$tab5['un'] = 12 ;  
$tab5['trois'] = "fraise" ;  
$tab5["deux"] = 2.5 ;  
$tab5[42] = "e15" ;
```

Clé	Valeur
"un"	12
"trois"	"fraise"
"deux"	2.5
42	"e15"

```
$tab6 = array('un' => 12,  
              'trois' => "fraise",  
              'deux' => 2.5,  
              42 => "e15") ;
```

Introduction au langage PHP

19

Syntaxe de base: Les Tableaux Associatifs

Parcours

```
foreach($tableau as $cle => $element)
{
    /* Bloc d'instructions répété pour
       chaque élément de $tableau */
    /* Chaque élément de $tableau est
       accessible grâce à $element */
    /* La clé d'accès à chaque élément est
       donnée par $cle */
}
```

```
<html>
  <head><title>foreach clé</title>
  </head>
  <body>

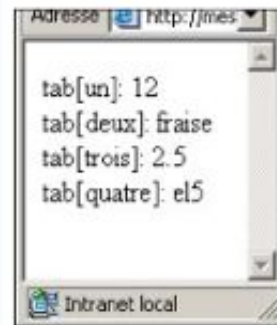
    <?php

      $tab6 = array('un'      => 12,
                    'deux'   => "fraise",
                    'trois'  => 2.5,
                    'quatre' => "el5");

      foreach ($tab6 as $cle => $val)
      { echo "tab[$cle]: $val<br>\n" ;
      }

    ?>

  </body></html>
```



Introduction au langage PHP

20

Syntaxe de base: Fonctions utilisateur

- Valeur de retour

 `function moyenne($a,$b)`
`{ ... }`

Typage faible de PHP :
Aucune information

- Arguments

`function moyenne(`  `$a,`  `$b)`
`{ ... }`

Typage faible de PHP :
Aucune information

Introduction au langage PHP

21

Syntaxe de base: Mode de passage des arguments

(types natifs)

```
<?php
function permutation($x, $y) {
    echo "permutation..." ;
    $t = $x ;
    $x = $y ;
    $y = $t ;
}
$a = 12 ;
$b = 210 ;
echo "\$a = $a" ;
echo "\$b = $b" ;
permutation($a, $b) ;
echo "\$a = $a" ;
echo "\$b = $b" ;
?>
```

Permutation impossible :
Passage des arguments
des fonctions par valeur

```
$a = 12
$b = 210
permutation...
$a = 12
$b = 210
```

Introduction au langage PHP

Syntaxe de base: Mode de passage des arguments
(types natifs)

```
<?php
function permutation(&$x, &$y) {
    echo "permutation..." ;
    $t = $x ;
    $x = $y ;
    $y = $t ;
}
$a = 12 ;
$b = 210 ;
echo "\$a = $a" ;
echo "\$b = $b" ;
permutation($a, $b) ;
echo "\$a = $a" ;
echo "\$b = $b" ;
?>
```

Permutation
réussie

\$a = 12
\$b = 210
permutation...
\$a = 210
\$b = 12

Introduction au langage PHP

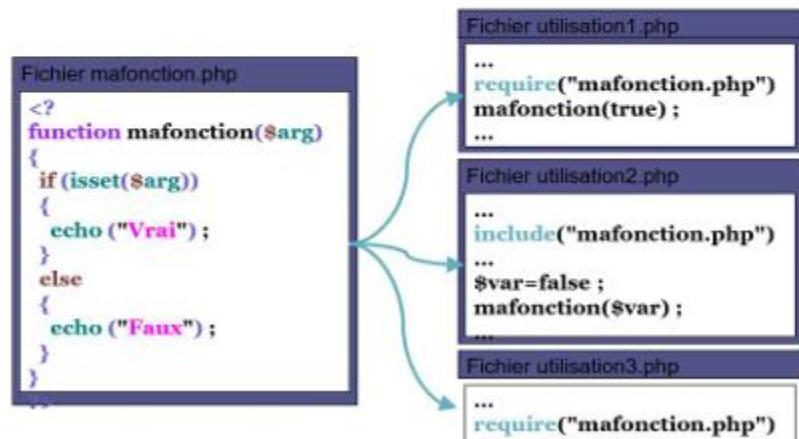
23

Pour pouvoir réutiliser certaines fonctions dans plusieurs scripts PHP

Utilisation de :

include et require

- `include("fichier") ;`
- `require("fichier") ;`
- `include_once("fichier") ;`
- `require_once("fichier") ;`



Traitement des données des formulaires en PHP

Traitement des données de formulaires

- Après le remplissage d'un formulaire HTML, les données sont traitées par une page PHP du serveur définie dans le champ **ACTION**.
- Une fois le serveur Web reçoit la requête (Get ou Post), les données sont contenues dans l'une des variables superglobales de type tableau associatif `$_GET` ou `$_POST`.
- La valeur de la clé du tableau porte le même nom indiqué lors de l'écriture des champs du formulaire de la page HTML de saisie.

Traitement des données des formulaires en PHP

Exemple – Formulaire HTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <title>formulaire</title>
</head>
<body>
  <form action="valide1.php"
    method="get">  Nom:
    <input type="text"
      name="nomPers">
    <input type="submit"
      value="Envoyer">
  </form>
</body>
</html>
```

Script PHP

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <title>Validation</title>
</head>
<body>
  <?php
    if (isset($_GET['nomPers']))
    {
      if (!empty($_GET['nomPers']))
      {
        echo "Vous avez saisi '"
          . $_GET['nomPers'] . "'\n" ;
      }
      else
        echo "Aucune valeur saisie\n";
    }
    else
      echo "Utilisation incorrecte\n" ;
  ?>
</body></html>
```

`$_GET['nomPers']`
est-il défini ?

`$_GET['nomPers']`
est-il vide ?

Traitement des données des formulaires en PHP

26

Formulaires contenant des champs SELECT

SELECT unique

```
<html>
<head>
<title> Formulaire de saisie des fruits</title>
</head>
<body>
  <form action="valide3.php"
  method="get"> Choisissez des
  fruits:&nbsp;
    <select name="sel">
      <option>Fraise
      <option> Pomme
      <option> Poire
      <option> Banane
      <option>Cerise
    </select>
    <input type="submit" value="envoyer">
  </form>
</body>
```

Traitement des données des formulaires en PHP

27

Formulaires contenant des champs SELECT SELECT multiple

```
<html>
<head>
  <title>Formulaire de saisie des fruits</title>
</head>
<body>
  <form action="valide3.php" method="get">
    Choisissez des fruits:&nbsp;
    <select name="sel[]" multiple>
      <option>Fraise
      <option>Pomme
      <option>Poire
      <option>Banane
      <option>Cerise
    </select>
    <input type="submit" value="envoyer">
  </form>
</body>
</html>
```

Envoyer

valide3.php?sel[]=Pomme&sel[]=Poire

Traitement des données des formulaires en PHP

28

Formulaires contenant des champs SELECT

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <title>Liste de fruits</title>
</head>
<body>

<?php
  if (isset($_GET['sel']) && !empty($_GET['sel']))
  { /* La variable $_GET['sel'] est définie
    et elle n'est pas vide */
    foreach($_GET['sel'] as $fruit)
      echo "Vous avez choisi $fruit<br>\n" ;
    }
  else
    echo "Vous n'avez pas choisi de fruit\n" ;
  ??
</body></html>
```

`$_GET['sel']`
est un tableau

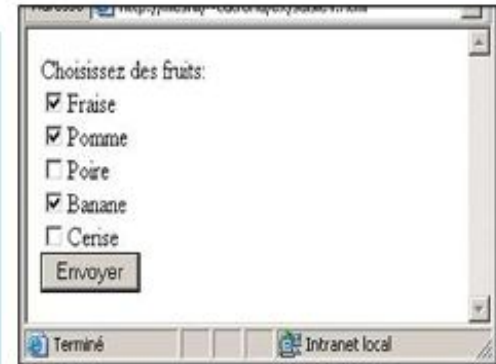


Traitement des données des formulaires en PHP

29

Formulaires contenant des champs CHECKBOX

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <title>Formulaire de saisie des fruits</title>
</head>
<body>
  <form name="formu" action="valide3.php"
  method="get"> Choisissez des fruits&nbsp;  <br>
  <input type="checkbox" name="sel[]" value="Fraise">Fraise<br>
  <input type="checkbox" name="sel[]" value="Pomme" >Pomme <br>
  <input type="checkbox" name="sel[]" value="Poire" >Poire <br>
  <input type="checkbox" name="sel[]" value="Banane">Banane<br>
  <input type="checkbox" name="sel[]" value="Cerise">Cerise<br>
  <input type="submit" value="Envoyer">
  </form>
</body>
</html>
```



Choisissez des fruits:

- ☒ Fraise
- ☒ Pomme
- ☐ Poire
- ☒ Banane
- ☐ Cerise

Envoyer

Terminé Intranet local



Adresse: http://localhost/valide3.php?sel[]=Fraise&sel[]=Pomme&sel[]=Banane

Vous avez choisi Fraise
Vous avez choisi Pomme
Vous avez choisi Banane

Principe

Les étapes nécessaire pour effectuer des requêtes vers une base de données:

1. Se connecter sur le serveur **MySQL**
2. Choisir la base de données de travail
3. Effectuer la requête
4. Récupérer et parcourir le résultat de la requête

Il existe des fonctions PHP immédiate permettant d'effectuer toutes ces opérations

Utilisation de MySQLi (Programmation Orientée Objet)

Spécifique à MySQL, plus rapide, et supporte les requêtes préparées.

```
$servername = "localhost";  
$username = "root";  
$password = "";  
$dbname = "ma_base";
```

1. Définition des paramètres de connexion

```
// Connexion à MySQL avec MySQLi  
$conn = new mysqli($servername, $username, $password, $dbname);
```

2. Connexion à MySQL avec MySQLi

```
// Vérifier la connexion  
if ($conn->connect_error) {  
    die("Échec de la connexion : " . $conn->connect_error);  
}
```

3. Vérification de la connexion

```
// Exécution d'une requête  
$sql = "SELECT * FROM utilisateurs";  
$result = $conn->query($sql);
```

4. Exécution d'une requête SQL

```
if ($result->num_rows > 0) {  
    while($row = $result->fetch_assoc()) {  
        echo "Nom : " . $row["nom"] . "<br>";  
    }  
} else {  
    echo "Aucun résultat trouvé.";  
}
```

5. Vérification et affichage des résultats

```
// Fermer la connexion  
$conn->close();
```

6. Fermeture de la connexion

Utilisation de PDO (PHP Data Objects)

Compatible avec plusieurs bases de données (MySQL, PostgreSQL, SQLite, ...), sécurisé et flexible

```
$dsn = "mysql:host=localhost;dbname=ma_base;charset=utf8mb4";  
$username = "root";  
$password = "";
```

1. Définition des paramètres de connexion

```
try {  
    $pdo = new PDO($dsn, $username, $password, [  
        PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION  
    ]);
```

2. Connexion à la base de données avec PDO

```
    // Requête SQL sécurisée avec requêtes préparées  
    $stmt = $pdo->prepare("SELECT * FROM utilisateurs WHERE email = :email");  
    $stmt->execute(["email" => "exemple@email.com"]);
```

3. Requête SQL sécurisée avec une requête préparée

```
    while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {  
        echo "Nom : " . $row["nom"] . "<br>";  
    }
```

4. Récupération et affichage des résultats

```
} catch (PDOException $e) {  
    echo "Erreur de connexion : " . $e->getMessage();  
}
```

5. Gestion des erreurs avec try...catch

Accès aux bases de données en PHP

35

Comparaison MySQLi vs PDO

Critère	MySQLi (Procédural)	MySQLi (Orienté Objet)	PDO
Style de code	Plus proche de MySQL	Plus structuré	Plus propre et flexible
Sécurité	Moins sécurisé (risque d'injection SQL sans précaution)	Sécurisé (requêtes préparées)	Très sécurisé (requêtes préparées par défaut)
Supporte plusieurs bases de données ?	✗ Non (Seulement MySQL)	✗ Non (Seulement MySQL)	✓ Oui (MySQL, PostgreSQL, SQLite, etc.)
Requêtes préparées	✓ Oui	✓ Oui	✓ Oui (avec <code>:param</code>)
Gestion des erreurs	<code>mysqli_error()</code>	<code>\$conn->error</code>	<code>PDOException</code>
Fermeture	<code>mysqli_close(\$conn)</code>	<code>\$conn->close()</code>	<code>\$pdo = null</code>

Accès aux bases de données en PHP

36

Comparaison des fonctions MySQLi vs PDO

Catégorie	MySQLi (Procédural)	MySQLi (Orienté Objet)	PDO
Connexion	<code>mysqli_connect(host, user, pass, db)</code>	<code>\$conn = new mysqli(host, user, pass, db);</code>	<code>\$pdo = new PDO("mysql:host=host;dbname=db", user, pass);</code>
Gestion des erreurs	<code>mysqli_connect_error()</code> ou <code>mysqli_error(\$conn)</code>	<code>\$conn->connect_error</code> ou <code>\$conn->error</code>	<code>try { } catch (PDOException \$e) { \$e->getMessage(); }</code>
Exécution d'une requête simple	<code>mysqli_query(\$conn, "SELECT * FROM users")</code>	<code>\$conn->query("SELECT * FROM users")</code>	<code>\$pdo->query("SELECT * FROM users")</code>
Requête préparée	<code>mysqli_prepare(\$conn, "SELECT * FROM users WHERE id = ?")</code>	<code>\$stmt = \$conn->prepare("SELECT * FROM users WHERE id = ?")</code>	<code>\$stmt = \$pdo->prepare("SELECT * FROM users WHERE id = :id")</code>

Accès aux bases de données en PHP

37

Comparaison des fonctions MySQLi vs PDO

Lier des paramètres	<code>mysqli_stmt_bind_param(\$stmt, "i", \$id)</code>	<code>\$stmt->bind_param("i", \$id)</code>	<code>\$stmt->bindParam(":id", \$id, PDO::PARAM_INT)</code>
Exécuter une requête préparée	<code>mysqli_stmt_execute(\$stmt)</code>	<code>\$stmt->execute()</code>	<code>\$stmt->execute()</code>
Récupérer les résultats (fetch_assoc)	<code>mysqli_fetch_assoc(\$result)</code>	<code>\$result->fetch_assoc()</code>	<code>\$stmt->fetch(PDO::FETCH_ASSOC)</code>
Récupérer les résultats (fetch_object)	<code>mysqli_fetch_object(\$result)</code>	<code>\$result->fetch_object()</code>	<code>\$stmt->fetch(PDO::FETCH_OBJ)</code>

Accès aux bases de données en PHP

38

Comparaison des fonctions MySQLi vs PDO

Boucler sur les résultats	<pre>while (\$row = mysqli_fetch_assoc(\$result)) { ... }</pre>	<pre>while (\$row = \$result- >fetch_assoc()) { ... }</pre>	<pre>while (\$row = \$stmt- >fetch(PDO::FETCH_ASSOC)) { ... }</pre>
Nombre de lignes retournées	<pre>mysqli_num_rows(\$result)</pre>	<pre>\$result- >num_rows</pre>	<pre>\$stmt->rowCount()</pre>
Dernier ID inséré	<pre>mysqli_insert_id(\$conn)</pre>	<pre>\$conn->insert_id</pre>	<pre>\$pdo->lastInsertId()</pre>
Libérer la mémoire	<pre>mysqli_free_result(\$result)</pre>	<pre>\$result->free()</pre>	<pre>\$stmt->closeCursor()</pre>
Fermeture de la connexion	<pre>mysqli_close(\$conn)</pre>	<pre>\$conn->close()</pre>	<pre>\$pdo = null;</pre>

Accès aux bases de données en PHP

39

Récupérer les résultats des requêtes en MySQLi (orienté objet) et PDO.

Type de récupération	MySQLi (Objet)	PDO
Tableau indexé	<code>\$row = \$result->fetch_row();</code>	<code>\$row = \$stmt->fetch(PDO::FETCH_NUM);</code>
Tableau associatif	<code>\$row = \$result->fetch_assoc();</code>	<code>\$row = \$stmt->fetch(PDO::FETCH_ASSOC);</code>
Mixte (indexé + associatif)	<code>\$row = \$result->fetch_array();</code>	<code>\$row = \$stmt->fetch(PDO::FETCH_BOTH);</code>
Objet PHP	<code>\$row = \$result->fetch_object();</code>	<code>\$row = \$stmt->fetch(PDO::FETCH_OBJ);</code>

Récupérer les résultats des requêtes en MySQLi

40


EXAMPLE

```
$conn = new mysqli("localhost", "root", "", "ma_base");

if ($conn->connect_error) {
    die("Échec de la connexion : " . $conn->connect_error);
}

$sql = "SELECT * FROM utilisateurs";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // Récupérer les résultats sous différentes formes
    while ($row = $result->fetch_row()) { // fetch_row() : tableau indexé
        echo "Nom : " . $row[1] . "<br>"; // Supposons que l'index 1 contient le nom
    }
}
```



Récupérer les résultats des requêtes en MySQLi

EXAMPLE

```
$result->data_seek(0); // Remettre Le pointeur au début

while ($row = $result->fetch_assoc()) { // fetch_assoc() : tableau associatif
    echo "Nom : " . $row["nom"] . "<br>";
}

$result->data_seek(0); // Remettre Le pointeur au début

while ($row = $result->fetch_array()) { // fetch_array() :
    echo "Nom : " . $row["nom"] . " (" . $row[1] . ")<br>";
}
```

Renvoie une ligne de résultat sous forme de tableau associatif, où les clés sont les noms des colonnes de la table

Renvoie une ligne de résultat à la fois sous forme de tableau indexé et associatif. Les colonnes peuvent être accessibles soit par leur index, soit par leur nom.

Récupérer les résultats des requêtes en MySQLi


42

EXAMPLE

```
$result->data_seek(0); // Remettre le pointeur au début

while ($row = $result->fetch_object()) { // fetch_object() : objet PHP
    echo "Nom : " . $row->nom . "<br>";
}
} else {
    echo "Aucun résultat trouvé.";
}

$conn->close();
```



Renvoie une ligne de résultat sous forme d'un objet PHP. Les propriétés de l'objet correspondent aux noms des colonnes de la table.

Récupérer les résultats des requêtes Avec PDO

EXAMPLE

```
$result->data_seek(0); // Remettre le pointeur au début

while ($row = $result->fetch_object()) { // fetch_object() : objet PHP
    echo "Nom : " . $row->nom . "<br>";
}
} else {
    echo "Aucun résultat trouvé.";
}

$conn->close();
```



Sessions & Cookies en PHP

44

Suite de chapitre III.....

Sessions & Cookies en PHP

45



Sessions & Cookies en PHP

46

