PHP PDO

Ce chapitre utilise quelques notions de POO. Le chapitre L'objet est donc un prérequis pour la bonne compréhension des syntaxes.

1. Introduction

PDO (*PHP Data Object*) est une bibliothèque de fonctions PHP permettant d'accéder à n'importe quelle base de données car elle apporte une couche d'abstraction à l'accès aux bases de données. C'est un langage orienté objet mais dont le fonctionnement n'est pas très différent de mysqli. Il faut toujours écrire la requête et l'exécuter.

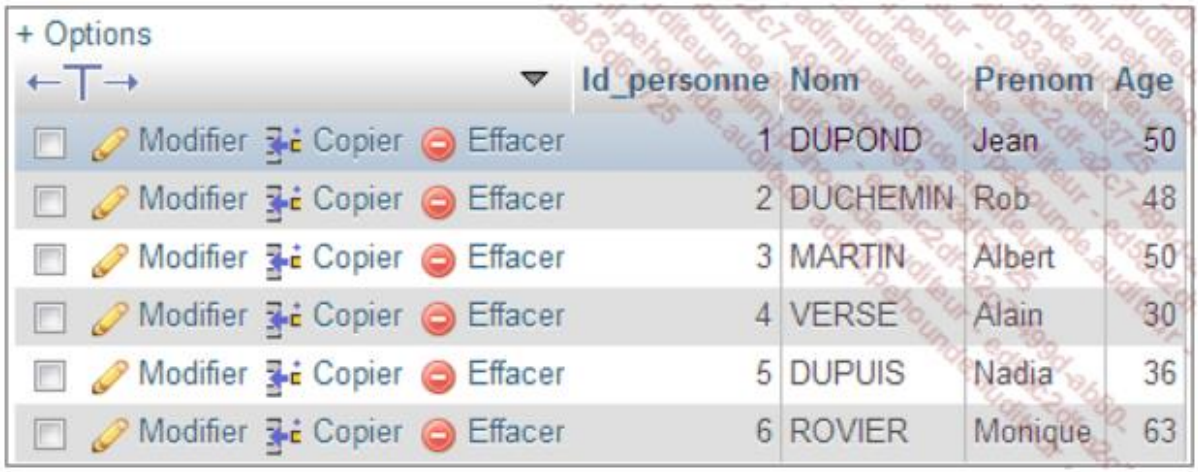
Pour activer cette bibliothèque, il faut ouvrir le fichier php.ini accessible par le menu configuration, PHP. Ensuite, vérifiez qu'il n'y a pas de point-virgule devant la ligne :

```
;extension=php_openssl.dll
;extension=php_pdo_firebird.dll
;extension=php_pdo_mssql.dll
extension=php_pdo_mysql.dll
;extension=php_pdo_oci.dll
;extension=php_pdo_odbc.dll
```

Cette bibliothèque est écrite en langage objet donc la syntaxe peut être déroutante. Vous avez plus d'explications sur la programmation objet dans le chapitre L'objet.

Il faut en prérequis avoir créé la table Personne.

Ainsi que ces données :



Le chapitre sur phpMyAdmin explique comment créer la base avec les données.

2. Connexion

Pour se connecter à la base de données MySQL, il faut créer une instance de la classe PDO, c'est-à-dire créer un objet qui est un exemplaire de la classe PDO mais avec certains paramètres. Cette notion est abordée de façon plus précise dans le chapitre L'objet.

L'objet permettant de se connecter à MySQL est `PDO()`.

Cet objet prend en paramètres :

- La chaîne de connexion : chaîne de caractères contenant le SGBD utilisé et le nom ou l'adresse IP de l'hôte. Celui-ci correspond à "localhost" ou 127.0.0.1 si vous travaillez en local. Cette chaîne contient aussi le nom de la base de données.
- L'utilisateur : chaîne de caractères contenant le nom de l'utilisateur pour se connecter à la base de données. Si vous travaillez en local, celui-ci correspond à "root". Attention, cet utilisateur a tous les droits sur votre base de données.
- Le mot de passe : chaîne de caractères contenant le mot de passe associé à l'utilisateur. Celui-ci est vide par défaut.

Par exemple, pour se connecter sur la base de données bdd :

```
<?php
$base = new PDO('mysql:host=127.0.0.1;dbname=bdd', 'root', '');
?>
```

La variable `$base` est un objet donc vous ne pouvez pas afficher sa valeur.

Pour tester si le code a généré une erreur, il faut entourer le code par l'instruction `try {} catch (Exception $e) {}`.

Par exemple :

```
<?php
try {
   $base = new PDO('mysql:host=127.0.0.1;dbname=bdd', 'root', '');
}
catch (Exception $e) {
   die('Erreur : ' . $e->getMessage());
}
?>
```

S'il se passe une erreur dans le bloc try, PHP passe automatiquement dans le bloc catch et donc exécute l'instruction `die()`. La fonction `die()` est équivalente à la fonction `exit()`, c'est-à-dire qu'elle termine le script courant en affichant un message.

Pour récupérer les erreurs lorsqu'elles se produisent, il faut ajouter le code suivant :

```
$base->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);
```

Ce code a pour effet d'activer les exceptions PDO.

Depuis la version 5.5 de PHP, il est possible d'utiliser le bloc `finally`. Ce bloc, placé après le `catch`, est toujours exécuté.

Par exemple :

```
<?php
try {
   $base = new PDO('mysql:host=127.0.0.1;dbname=bdd', 'root', '');
   $base->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);
   echo "Connexion ok.";
}
catch (Exception $e) {
   die('Erreur : ' . $e->getMessage());
}

finally {

   $base = null; //fermeture de la connexion

}
?>
```

Affiche :

Connexion ok.

Dans tous les cas, le code passe dans le bloc finally et ferme la connexion.

Pour se connecter à SQL Server, il faut installer la bibliothèque php_pdo_sqlsrv_55_ts.dll (la version 7 n'existe pas au moment de l'écriture) en copiant ce fichier dans le dossier ext. Puis il faut ajouter extension=php_pdo_sqlsrv_55_ts.dll dans php.ini. Enfin, il faut installer Microsoft ODBC Driver 11 for SQL Server. La syntaxe est :

```
<?php
try {
   $base = new PDO("sqlsrv:Server=SERVEUR;Database=Nom_base",
"login_base", "pw_base"););
   $base->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);
   echo "Connexion ok.";
}
```

```
catch (Exception $e) {
   die('Erreur : ' . $e->getMessage());
}

finally {

   $base = null; //fermeture de la connexion

}
?>
```

Affiche :

Connexion ok.

3. Requêtes non préparées

a. Lire des données

La méthode permettant d'exécuter une requête SQL est `query()`.

Cette méthode fait partie de l'objet connexion renvoyé par `new PDO()`. Elle prend en paramètre la requête SQL sous forme de chaîne de caractères.

Cette fonction retourne un objet resultat contenant tout ce que renvoie la requête SQL.

La méthode permettant de connaître le nombre de lignes dans le résultat de la requête est `rowCount()`.

Cette méthode fait partie de l'objet `$resultat` renvoyé par la méthode `query()`.

Par exemple, pour avoir le nombre de lignes dans la table Personne :

```
<?php
try
{
   $base = new PDO('mysql:host=127.0.0.1;dbname=bdd', 'root', '');
   $base->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);

   // Récupération des données de la table Personne
   $resultat = $base->query('SELECT * FROM Personne');

   echo "Nombre de personnes : ".$resultat->rowCount();
   // Affichage du nombre de lignes
```

```
   $resultat->closeCursor(); // Fermeture de la requête
}
catch(Exception $e)
{
   // message en cas d'erreur
   die('Erreur : '.$e->getMessage());
}
?>
```

Affiche :

Nombre de personnes : 6

Maintenant, si vous voulez afficher les données de la table Personne, il faut utiliser la notion de fetch. Le fetch permet de lire la ligne courante et de se déplacer sur la ligne suivante.

La méthode permettant d'effectuer le fetch de la requête est `fetch()`.

Cette méthode fait partie de l'objet `$resultat` renvoyé par la méthode `query()`.

Par exemple, pour afficher le nom et le prénom de la table Personne :

```
<?php
try
{
   $base = new PDO('mysql:host=127.0.0.1;dbname=bdd', 'root', '');
   $base->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);

   // Récupération des données de la table Personne
   $resultat = $base->query('SELECT Nom, Prenom FROM Personne');

   echo "Nombre de personnes : ".$resultat->rowCount();
   // Affichage de chaque entrée une à une
   while ($donnees = $resultat->fetch())
   {
   ?>
      <p>
      Nom : <?php echo $donnees['Nom']; ?>,
      Prénom : <?php echo $donnees['Prenom']; ?>
      </p>
   <?php
   }
```

```
   $resultat->closeCursor(); // Fermeture de la requête
}
catch(Exception $e)
{
   // message en cas d'erreur
   die('Erreur : '.$e->getMessage());
}


?>
```

Affiche :

Nombre de personnes : 6
Nom : DUPOND, Prénom : Jean
Nom : DUCHEMIN, Prénom : Rob
Nom : MARTIN, Prénom : Albert
Nom : VERSE, Prénom : Alain
Nom : DUPUIS, Prénom : Nadia
Nom : ROVIER, Prénom : Monique

b. Écrire des données

Pour écrire des données, il faut exécuter une requête de type INSERT.

La méthode permettant d'exécuter une requête SQL de type UPDATE, INSERT ou DELETE est `exec()`.

Cette méthode fait partie de l'objet connexion renvoyé par `new PDO()`. Elle prend en paramètre la requête SQL sous forme de chaîne de caractères.

Par exemple, pour insérer une personne nommée Olivier DURAND âgée de 36 ans :

```
<?php


try
{
  $base = new PDO('mysql:host=127.0.0.1;dbname=bdd', 'root', '');
  $base->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);


  $sql = "INSERT INTO Personne (Nom, Prenom, Age) VALUES
('DURAND','Olivier',36)";
   // Ajout des données dans la table Personne
   $base->exec($sql);
   echo "Personne ajoutée.";
}
```

```
catch(Exception $e)
{
   // message en cas d'erreur
   die('Erreur : '.$e->getMessage());
}

?>
```

Vous avez alors une nouvelle ligne dans la base de données :

Vous remarquez que l'Id_personne n'a pas été ajouté dans la requête. En effet, il est auto-incrémental donc c'est la base de données qui va automatiquement affecter un nouveau nombre à l'Id_personne.

Donc lorsque vous insérez une nouvelle personne, vous ne connaissez pas son identifiant. Pour récupérer le dernier Id auto-incrémental ajouté dans la base de données, il faut utiliser la méthode `lastInsertId()`. Cette méthode fait partie de l'objet connexion renvoyé par `new PDO()` et retourne le dernier entier auto-incrémental ajouté en base de données.

Par exemple, pour insérer une personne nommée Gérard ROLLAND âgée de 64 ans :

```
<?php
try
{
   $base = new PDO('mysql:host=127.0.0.1;dbname=bdd', 'root', '');
   $base->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);

   $sql = "INSERT INTO Personne (Nom, Prenom, Age) VALUES
('ROLLAND','Gérard',64)";
   // Ajout des données dans la table Personne
   $base->exec($sql);
   echo "L'identifiant de la dernière personne ajoutée est : ";
   echo $base->lastInsertId().".";
}
catch(Exception $e)
{
   // message en cas d'erreur
   die('Erreur : '.$e->getMessage());
}

?>
```

Affiche :

L'identifiant de la dernière personne ajoutée est : 8.

c. Supprimer des données

Pour supprimer des données, il faut exécuter une requête de type DELETE.

La méthode permettant d'exécuter une requête SQL de type UPDATE, INSERT ou DELETE est `exec()`.

Cette méthode fait partie de l'objet connexion renvoyé par `new PDO()`. Elle prend en paramètre la requête SQL sous forme de chaîne de caractères.

Par exemple, pour supprimer une personne dont le nom est VERSE :

```
<?php


try
{
   $base = new PDO('mysql:host=127.0.0.1;dbname=bdd', 'root', '');
   $base->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);


   $sql = "DELETE FROM Personne WHERE Nom = 'VERSE'";
   // Suppression des données dans la table Personne
   $base->exec($sql);
   echo "Personne supprimée.";
}
catch(Exception $e)
{
   // message en cas d'erreur
   die('Erreur : '.$e->getMessage());
}


?>
```

Affiche :

Personne supprimée.

d. Mettre à jour des données

Pour modifier des données, il faut exécuter une requête de type UPDATE.

La méthode permettant d'exécuter une requête SQL de type UPDATE, INSERT ou DELETE est : `exec()`.

Cette méthode fait partie de l'objet connexion renvoyé par `new PDO()`. Elle prend en paramètre la requête SQL sous forme de chaîne de caractères.

Pour avoir le nombre de lignes modifiées sur une requête de type UPDATE, il faut utiliser le retour de la méthode `exec()`. Celle-ci retourne le nombre de lignes affectées.

Par exemple, pour modifier la personne dont le nom est DURAND en changeant son nom par DUPUIS et son âge de 36 à 33 :

```
<?php


try
{
   $base = new PDO('mysql:host=127.0.0.1;dbname=bdd', 'root', '');
   $base->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);


  $sql = "UPDATE Personne SET Nom = 'DUPUIS', Age = 33  WHERE Nom =
'DURAND'";
   // Modification des données dans la table Personne
   $nombre = $base->exec($sql);
   echo "Nombre de personnes modifiées : ".$nombre;
}
catch(Exception $e)
{
   // message en cas d'erreur
   die('Erreur : '.$e->getMessage());
}


?>
```

Affiche :

Nombre de personnes modifiées : 1

4. Requêtes préparées

a. Lire des données

La méthode permettant de préparer une requête SQL de type SELECT, UPDATE, DELETE ou INSERT est `prepare()`.

Cette méthode fait partie de l'objet connexion renvoyé par `new PDO()`. Elle prend en paramètre une chaîne de caractères contenant la requête SQL avec des noms ou des marqueurs `?` à lier à différentes valeurs.

Cette fonction retourne un objet de type PDOStatement contenant tout ce que renvoie la requête SQL.

Par exemple, pour afficher le nom et le prénom de la table Personne qui ont un âge supérieur à 35 ans et le nom commençant par 'DU' :

```
<?php


try
{
   $base = new PDO('mysql:host=127.0.0.1;dbname=bdd', 'root', '');
   $base->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);


   $sql = "SELECT Nom, Prenom FROM Personne WHERE Age > ? AND Nom LIKE ?";
   // Préparation de la requête avec les marqueurs
   $resultat = $base->prepare($sql);
   $resultat->execute(array(35,'DU%')); //execute prend en paramètre
   un tableau contenant les valeurs dans l'ordre lorsque le marqueur
   est un ?
   while ($ligne = $resultat->fetch())
   {
      echo 'Nom : '.$ligne['Nom'].', Prénom : '.$ligne['Prenom'].'<br />';
   }


   $resultat->closeCursor();


}
catch(Exception $e)
{
   // message en cas d'erreur
   die('Erreur : '.$e->getMessage());
}
?>
```

Affiche :

Nom : DUPOND, Prénom : Jean
Nom : DUCHEMIN, Prénom : Rob
Nom : DUPUIS, Prénom : Nadia

Le même exemple en nommant les marqueurs :

```
<?php

try
{
   $base = new PDO('mysql:host=127.0.0.1;dbname=bdd', 'root', '');
   $base->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);

   $sql = "SELECT Nom, Prenom FROM Personne WHERE Age > :age AND
Nom LIKE :nom";
   // Préparation de la requête avec les marqueurs
   $resultat = $base->prepare($sql);
   $resultat->execute(array('age' => 35,'nom' => 'DU%'));
   //execute prend en paramètre un tableau contenant comme clé le nom
   des marqueurs ainsi que les valeurs correspondantes
   while ($ligne = $resultat->fetch())
   {
     echo 'Nom : '.$ligne['Nom'].', Prénom : '.$ligne['Prenom'].'<br />';
   }

   $resultat->closeCursor();

}
catch(Exception $e)
{
   // message en cas d'erreur
   die('Erreur : '.$e->getMessage());
}
?>
```

Affiche :

Nom : DUPOND, Prénom : Jean
Nom : DUCHEMIN, Prénom : Rob
Nom : DUPUIS, Prénom : Nadia

Cette solution a l'avantage d'être un peu plus lisible dans les paramètres passés à la requête SQL.

b. Écrire des données

De la même façon que pour lire des données, vous allez utiliser les méthodes `prepare()` et `execute()`.

Par exemple, pour insérer une personne nommée Claire ROLIN âgée de 42 ans :

```
<?php


try
{
   $base = new PDO('mysql:host=127.0.0.1;dbname=bdd', 'root', '');
   $base->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);


   $sql = "INSERT INTO Personne (Nom, Prenom, Age) VALUES (:nom,
:prenom, :age)";
   // Préparation de la requête avec les marqueurs
   $resultat = $base->prepare($sql);
   $resultat->execute(array('nom' => 'ROLIN','prenom' => 'Claire',
'age' => 42));
   echo "L'identifiant de la dernière personne ajoutée est : ";
   echo $base->lastInsertId().".";
   $resultat->closeCursor();


}
catch(Exception $e)
{
   // message en cas d'erreur
   die('Erreur : '.$e->getMessage());
}


?>
```

Affiche :

L'identifiant de la dernière personne ajoutée est : 9.

Dans phpMyAdmin, vous pouvez voir :

Dans le cas où vous souhaitez exécuter plusieurs fois de suite une requête d'insertion, il est intéressant d'utiliser la liaison des paramètres grâce à la méthode `bindParam()`.

Par exemple, pour insérer une personne nommée Jean MARTINEAU âgée de 57 ans ainsi qu'une autre personne nommée Bob MARTINET âgée de 45 ans :

```php
<?php
try
{
   $base = new PDO('mysql:host=127.0.0.1;dbname=bdd', 'root', '');
   $base->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);

  $sql = "INSERT INTO Personne (Nom, Prenom, Age) VALUES (:nom,
:prenom, :age)";
   // Préparation de la requête avec les marqueurs
   $resultat = $base->prepare($sql);
   // Liaison des paramètres.
   $resultat->bindParam(':nom',$nom);
   $resultat->bindParam(':prenom',$prenom);
   $resultat->bindParam(':age',$age);
   //première personne
   $nom = "MARTINEAU";
   $prenom = "Jean";
   $age = 57;
   $resultat->execute();
   echo "L'identifiant de la dernière personne ajoutée est : ";
   echo $base->lastInsertId().".<br />";
   //deuxième personne
   $nom = "MARTINET";
   $prenom = "Bob";
   $age = 45;
   $resultat->execute();
   echo "L'identifiant de la dernière personne ajoutée est : ";
   echo $base->lastInsertId().".";

   $resultat->closeCursor();

}
catch(Exception $e)
{
   // message en cas d'erreur
   die('Erreur : '.$e->getMessage());
}

?>
```

Affiche :

L'identifiant de la dernière personne ajoutée est : 10.
L'identifiant de la dernière personne ajoutée est : 11.

c. Supprimer des données

De la même façon que pour écrire ou lire des données, vous allez utiliser les méthodes `prepare()` et `execute()`.

Par exemple, pour supprimer MARTIN :

```
<?php


try
{
   $base = new PDO('mysql:host=127.0.0.1;dbname=bdd', 'root', '');
   $base->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);


   $sql = "DELETE FROM Personne WHERE Nom =:nom";
   // Préparation de la requête avec les marqueurs
   $resultat = $base->prepare($sql);
   $resultat->execute(array('nom' => 'MARTIN'));
   echo "Personne supprimée.";
   $resultat->closeCursor();


}
catch(Exception $e)
{
   // message en cas d'erreur
   die('Erreur : '.$e->getMessage());
}
?>
```

Affiche :

Personne supprimée.

d. Modifier des données

De la même façon que pour écrire ou lire des données, vous allez utiliser les méthodes `prepare()` et `execute()`.

Par exemple, pour modifier l'âge de Jean DUPOND de 50 à 55 :

```php
<?php
try
{
   $base = new PDO('mysql:host=127.0.0.1;dbname=bdd', 'root', '');
   $base->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);

   $sql = "UPDATE Personne SET Age = :age WHERE Nom = :nom AND
Prenom = :prenom";
   // Préparation de la requête avec les marqueurs
   $resultat = $base->prepare($sql);
   $resultat->execute(array('age' => 55, 'nom' => 'DUPOND',
'prenom' => 'Jean'));
   echo "Personne modifiée.";
   $resultat->closeCursor();

}
catch(Exception $e)
{
   // message en cas d'erreur
   die('Erreur : '.$e->getMessage());
}

?>
```

Affiche :

Personne modifiée.

e. Appeler une procédure stockée

Une procédure stockée possède un nom et peut avoir des paramètres d'entrée ou de sortie.

Des explications plus détaillées sont données dans ce chapitre, section SQL avancé - Les procédures stockées et les fonctions.

Il faut préalablement avoir créé les procédures stockées `creation_personne` et `cube` de la section SQL avancé - Les procédures stockées et les fonctions. Vérifiez que le champ Id_langue n'est pas dans la procédure stockée `creation_personne`.

La première procédure stockée `creation_personne` n'a que des paramètres d'entrée. Cette procédure s'appelle avec le mot-clé `CALL` puis il suffit de lier ses paramètres avec la fonction `bindParam()` et d 'exécuter la requête.

Par exemple, pour ajouter la personne Gilles DEVERS ayant 40 ans :

```php
<?php
try
{
   // Connexion à la base de données
   $base = new PDO('mysql:host=127.0.0.1;dbname=bdd', 'root', '');
   $base->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);

   $stmt = $base->prepare("CALL creation_personne(?,?,?)");
   $nom = 'DEVERS';
   $prenom = 'Gilles';
   $age = 40;
   $stmt->bindParam(1, $nom, PDO::PARAM_STR, 20); //paramètre de type
                                                  //STRING
   $stmt->bindParam(2, $prenom, PDO::PARAM_STR, 20);
   $stmt->bindParam(3, $age, PDO::PARAM_INT); //paramètre de type
   INTEGER

   // Exécution de la procédure stockée
   $stmt->execute();
   echo "La procédure a inséré une nouvelle personne.";
}
catch(Exception $e)
{
   // message en cas d'erreur
   die('Erreur : '.$e->getMessage());
}
?>
```

Affiche :

La procédure a inséré une nouvelle personne.

La deuxième procédure stockée `cube` a un paramètre d'entrée et un paramètre de sortie.

Cette procédure renvoie la valeur en entrée au cube. Le problème est qu'il existe un bug PDO/MySQL empêchant de récupérer la valeur de retour de façon standard. Il faut donc exécuter une deuxième requête pour récupérer la valeur de sortie.

```php
<?php
try
{
   // Connexion à la base de données
```

```
   $base = new PDO('mysql:host=127.0.0.1;dbname=bdd', 'root', '');
   $base->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);

   //appel de la procédure stockée et de la requête récupérant
   //le paramètre de sortie.
   $statement = $base->prepare('CALL cube(:entree, @sortie);
SELECT @sortie AS sortie;');
   $entree = 3;
   $statement->bindParam(':entree', $entree);
   //exécution de la procédure stockée
   $statement->execute();

   $statement->nextRowset();
   //lecture du paramètre de sortie
   $row = $statement->fetchObject();
   echo "La valeur ".$entree." au cube est : ".$row->sortie;

}
catch(Exception $e)
{
   // message en cas d'erreur
   die('Erreur : '.$e->getMessage());
}

?>
```

Affiche :

La valeur 3 au cube est : 27