

# PDO 1/3

Auteur : Sébastien Inion

studi

Photo de Kritsada Seekham: <https://www.pexels.com/fr-fr/photo/nature-eau-bleu-animal-7836299/>

[https://github.com/SebInfo/PDO\\_Studi\\_Part1](https://github.com/SebInfo/PDO_Studi_Part1)

TOUT CE QUE VOUS AVEZ TOUJOURS  
VOULU SAVOIR SUR PDO SANS  
JAMAIS OSER LE DEMANDER !

# PRINCIPE DE FONCTIONNEMENT DU PDO

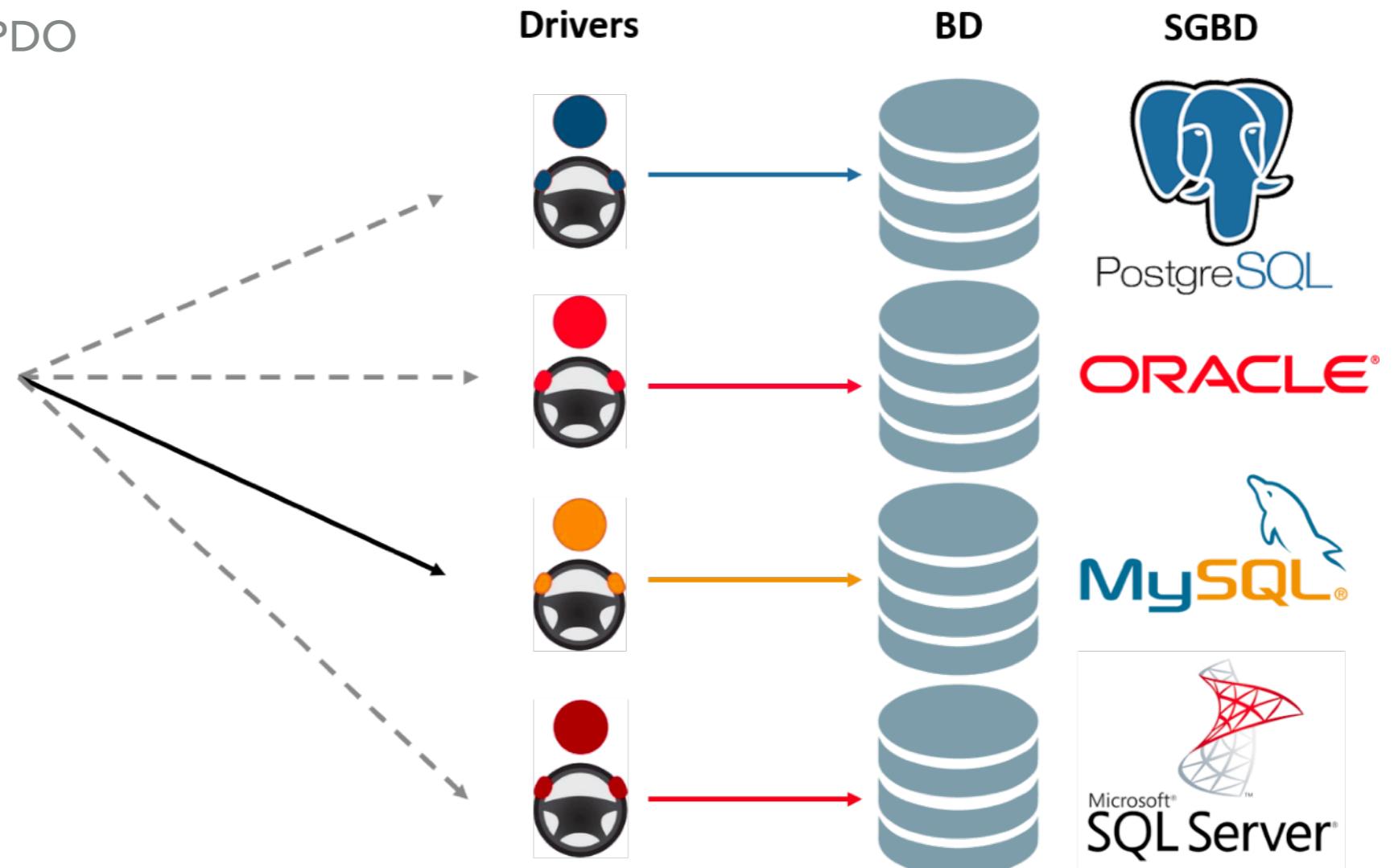


# UTILISER UNE BASE DE DONNÉES AVEC PHP

Architecture des drivers PDO



**PDO**  
(PHP Data Objects)



# PDO - PHP DATA OBJECTS

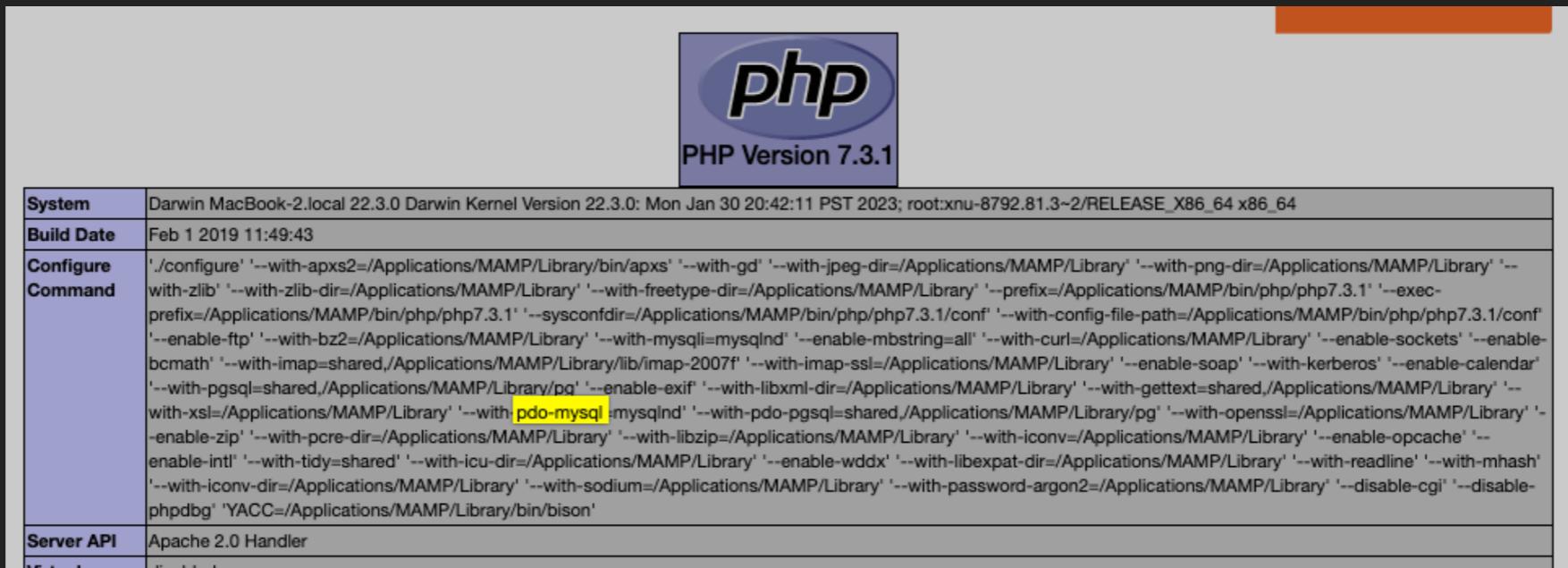
- ▶ PDO permet d'accéder à presque tous les SGBD : MySQL, Oracle, PostgreSQL, IBM DB2, SQLite ...
- ▶ Il présente une syntaxe orientée objet.
- ▶ C'est une extension qui définit une interface d'abstraction
- ▶ PDO ne fournit pas d'abstraction de base de données
  - ▶ SQL spécifique au moteur
  - ▶ Fonctionnalités présentes ou absentes suivant le moteur (SGBD)
- ▶ L'avantage d'une interface est une indépendance par rapport au SGBD et aux changements éventuels.
- ▶ PDO ne fonctionne qu'à partir de la version 5 (PHP 5)
- ▶ L'extension est activé par défaut à partir de 5.1.0
- ▶ Pour chaque SGBD on a un pilote.

# PDO - ACTIVATION DE PDO ET DU DRIVER SGBD

PDO

PDO support	enabled
PDO drivers	mysql, sqlite

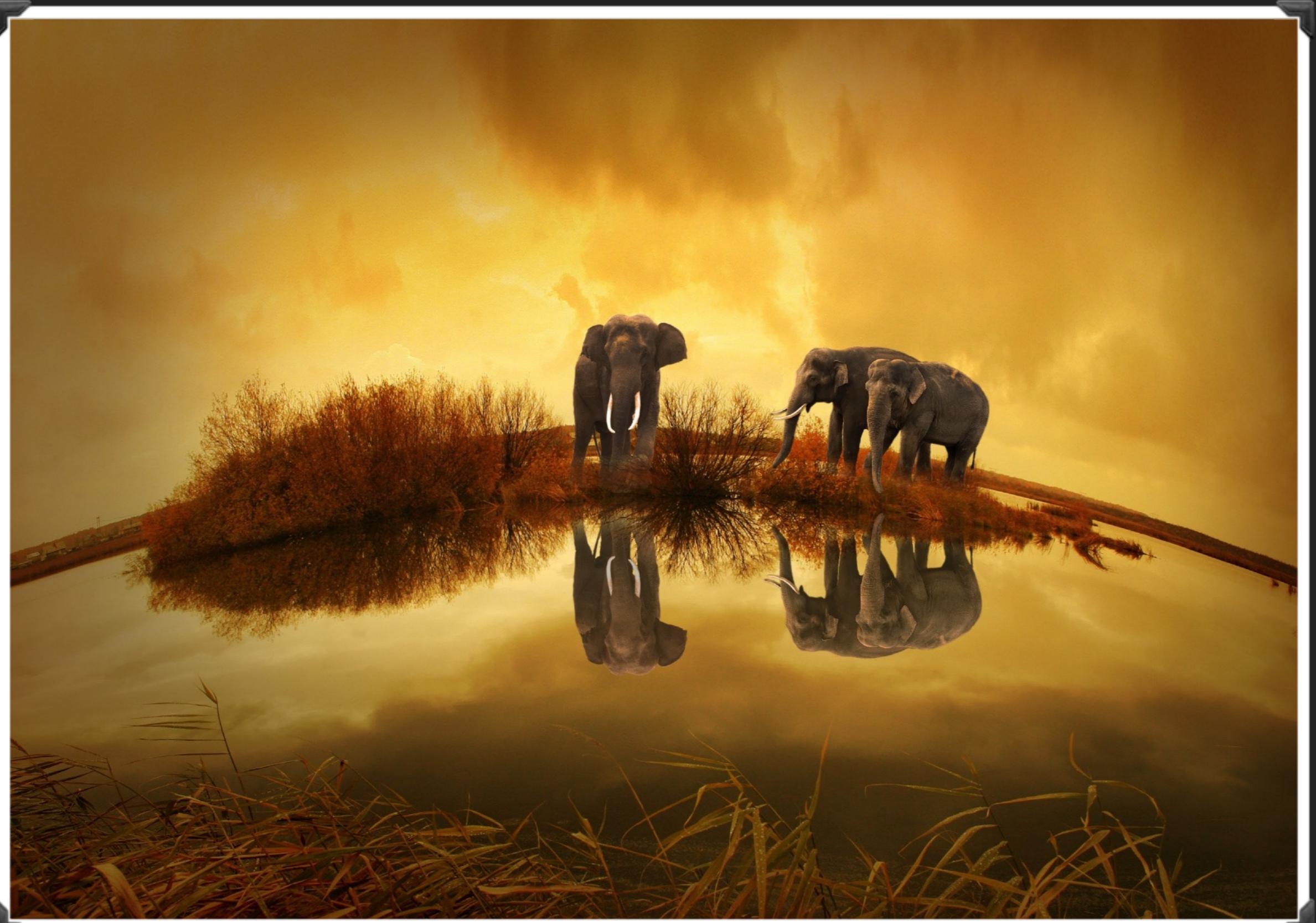
- La configuration prend en compte nativement MySQL et MariaDB.



- Si besoin il faut modifier le fichier php.ini et décommenter la ligne qui correspond à votre SGBD

```
;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
;extension=php_pdo_pgsql.dll
;extension=php_pdo_sqlite.dll
;extension=php_pdo_sql.dll
```

# PRÉSENTATION DES CLASSES PDO



## PDO - PHP DATA OBJECTS

- ▶ PDO c'est en fait 3 classes :
  - ▶ la classe **PDO**
  - ▶ la classe **PDOStatement**
  - ▶ la classe **PDOException**

## LA CLASSE PDO

- ▶ La classe **PDO** : gère une connexion entre PHP et un serveur de base de données.
- ▶ Gestion des transactions (sessions)
- ▶ Gestion des erreurs
- ▶ Prépare les requêtes
- ▶ Exécute des requêtes sans préparation
- ▶ Paramétrage des attributs PDO

PDO
+__construct()
+beginTransaction()
+commit()
+errorCode()
+errorInfo()
+exec()
+getAttribute()
+getInserId()
+prepare()
+query()
+quote()
+rollBack()
+setAttribute()

## LA CLASSE PDOStatement

- ▶ La classe **PDOStatement** : représente une requête préparée et, une fois exécutée, le jeu de résultats associé.
- ▶ Gestion des erreurs
  - ▶ Associe valeurs et paramètres
  - ▶ Exécute des requêtes préparées
  - ▶ Récupère les erreurs des requêtes
- ▶ il n'y a pas de constructeur car c'est la classe PDO qui va créer les objets **PDOStatement** avec certaines méthodes.

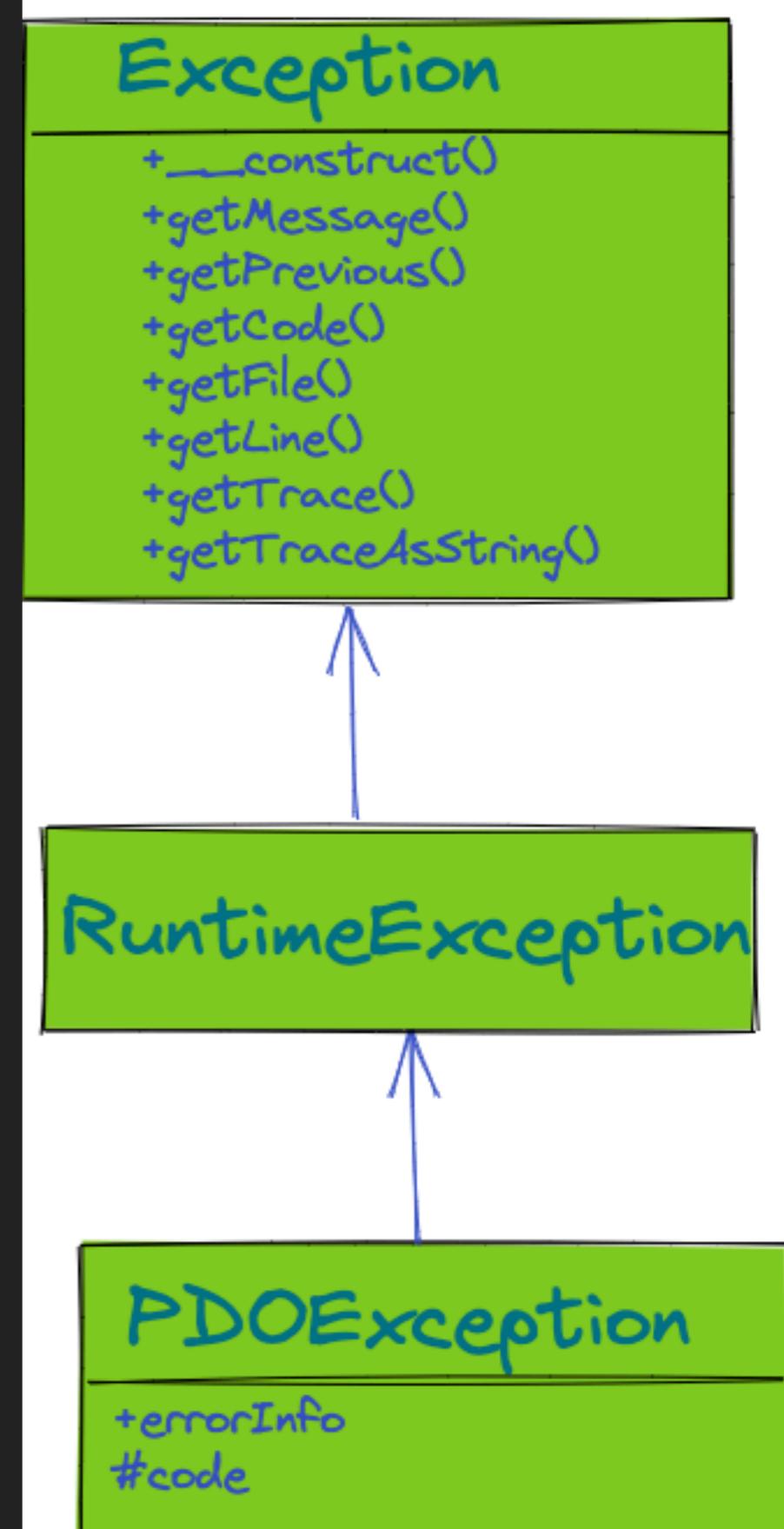
### PDOStatement

```
+bindColumn()  
+bindParam()  
+bindValue()  
+errorCode()  
+errorInfo()  
+execute()  
+fetch()  
+fetchAll()  
+fetchColumn()  
+getAttribute()  
+rowCount()  
+setAttribute()  
+setFetchMode()
```

## LA CLASSE PDOEXCEPTION

- ▶ La classe **PDOException** : Représente une erreur (exception) émise par PDO. Cette classe gère les erreurs.
- ▶ Classe personnalisée destinée à émettre une exception
- ▶ Vous ne devez pas lancer une exception **PDOException** depuis votre propre code.
- ▶ Elle est lancée par **PDO** ou **PDOStatement**

<https://www.php.net/manual/fr/class pdoexception>

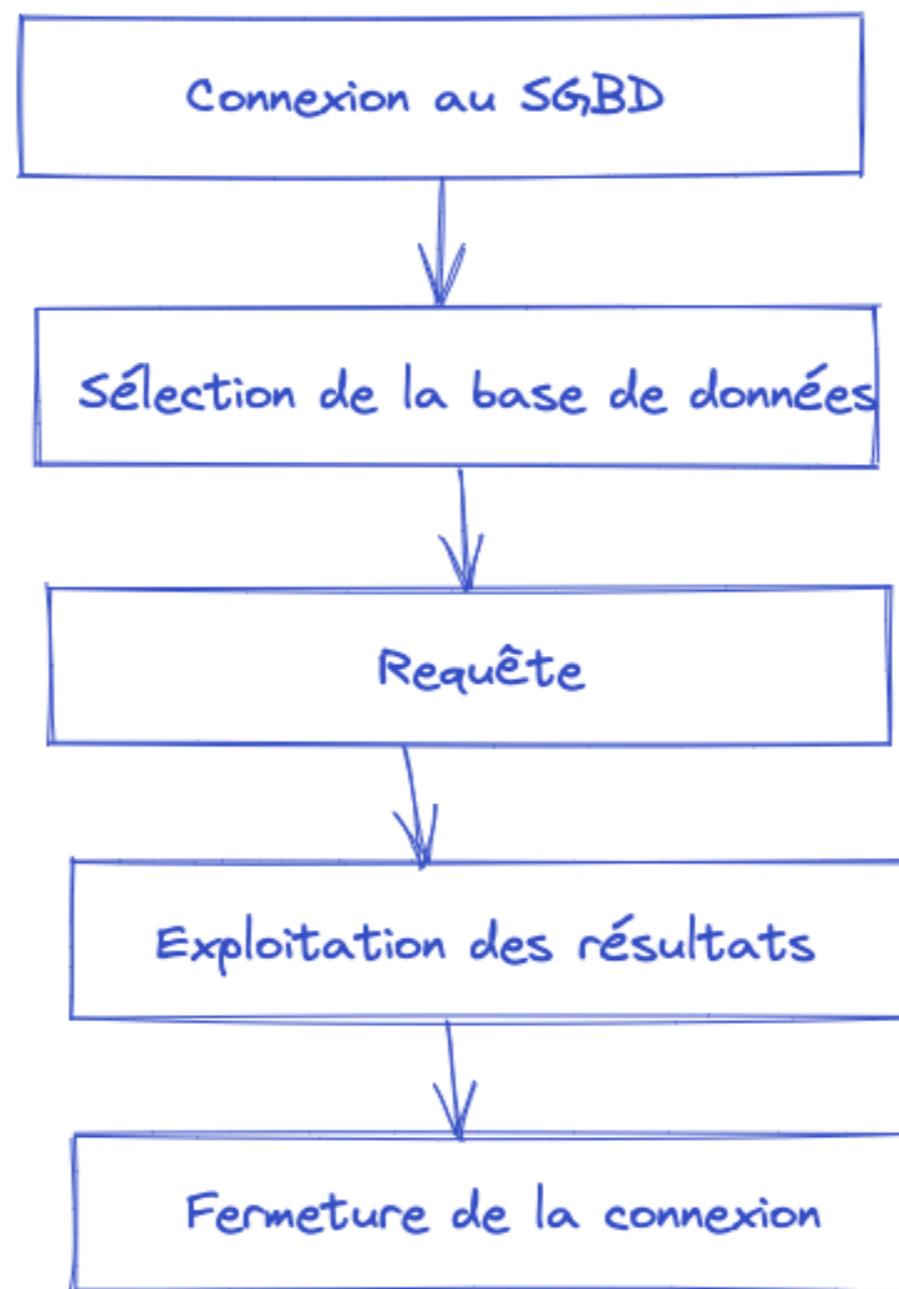


# CONNEXION AVEC PDO ET GESTION DES ERREURS



# UTILISER UNE BASE DE DONNÉES

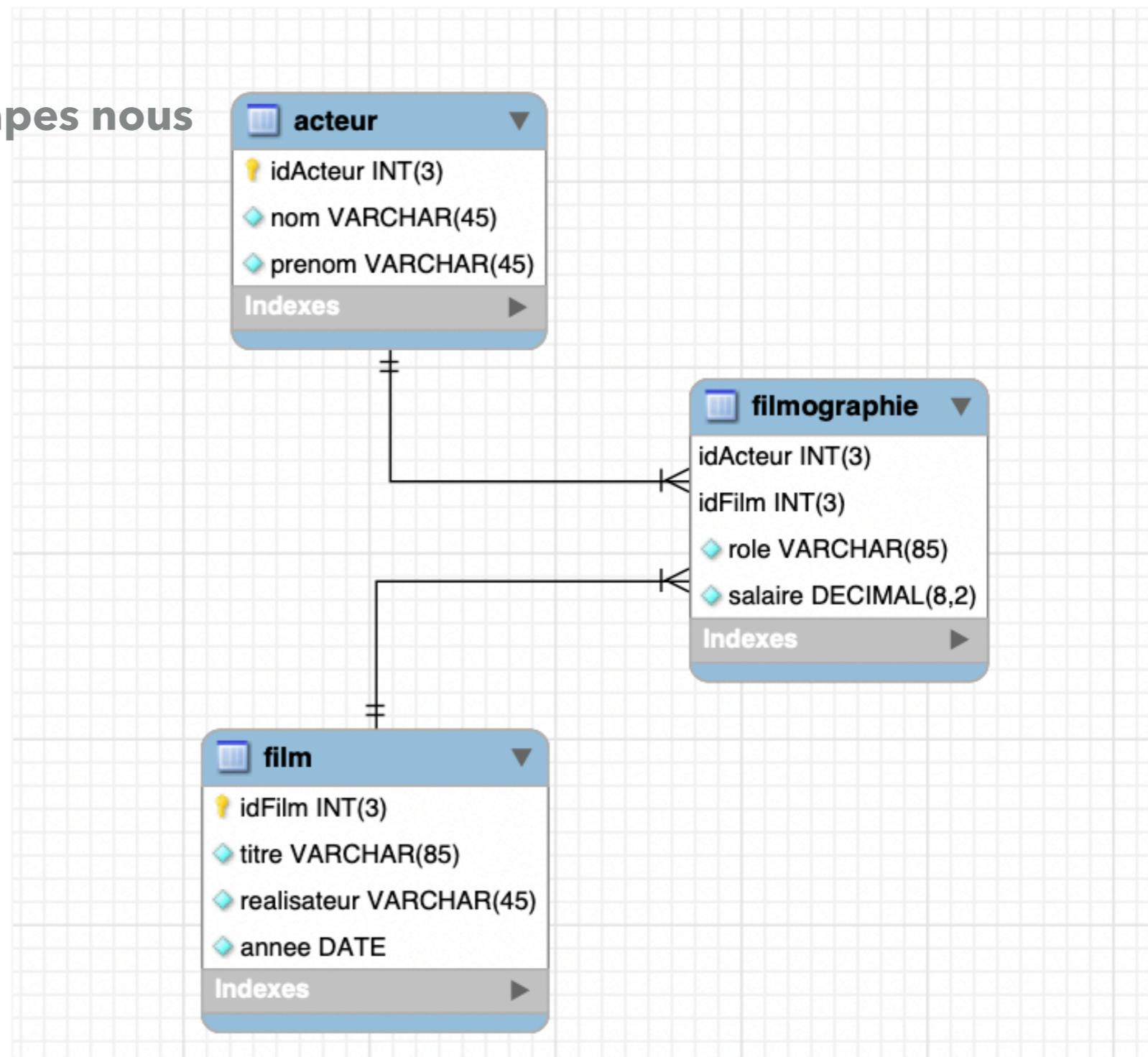
Les cinq étapes pour accéder à vos données



**Remarque :** la dernière action n'est pas obligatoire puisque quand le script PHP se termine la connexion est fermée automatiquement. Cela dit garder ouverte une connexion peut consommer des ressources inutilement.

# MISE EN PRATIQUE DE CES 5 ÉTAPES AVEC LA BASE FILMS

- Pour illustrer les différentes étapes nous utiliserons la base **FILMS**
- Qui ne comporte que 3 tables
  - film
  - acteur
  - filmographie



## LE CONTENU DE LA BASE FILMS

```
1 • SELECT * FROM FILMS.acteur;
```

100% 1:1

Result Grid Filter Rows: Search Edit:

idActeur	nom	prenom
1	Johnny	Deep
2	Al	Pacino
3	Suraj	Sharma
4	Brad	Pitt
5	Edward	Norton
6	D'artagnan	Edourd

```
1 • SELECT * FROM FILMS.filmographie;
```

100% 1:1

Result Grid Filter Rows: Search Edit:

idActeur	idFilm	role	salaire
1	5	Roux	5000.00
1	7	Rango	10000.00
2	2	Michael Corleone	10000.00
2	3	Michael Corleone	20000.00
2	6	Tony Montana	15000.00
3	4	Pi	20000.00

Lien pour récupérer le DUMP qui permettra de créer la base :

[https://github.com/SebInfo/PDO\\_Studi\\_Part1](https://github.com/SebInfo/PDO_Studi_Part1)

```
1 • SELECT * FROM FILMS.film;
```

100% 1:1

Result Grid Filter Rows: Search Edit:

idFilm	titre	realisateur	annee
1	Les évadés	Darabont	1994-00-00
2	Le parrain	Coppola	1972-00-00
3	Le parrain 2	Coppola	1974-00-00
4	L'odyssée de Pi	Ang Lee	2013-00-00
5	Chocolat	Hallstrom	2000-00-00
6	Scarface	De Palma	1983-00-00
7	Rango	Verbinski	2011-00-00

# CONNEXION À LA BASE DE DONNÉES

- Création d'un objet PDO avec l'instruction `new()`
- Le constructeur prend 4 paramètres
  - le **DSN (Data Source Name)** est un paramètre obligatoire
    - On indique le driver **mysql**
    - Où se trouve le serveur : ici en local donc **127.0.0.1** ou **localhost**
    - On peut indiquer le port avec **port = N° du port**
    - On peut indiquer la base de données à laquelle on veut se connecter avec **dbname = nom\_de\_la\_base**
  - le **nom de l'utilisateur** (facultatif si non nécessaire)
  - le **mot de passe** (facultatif si il y a pas de mot de passe)
  - **Options (falcultatif): C'est un tableau clé=>valeur avec les options spécifiques de connexion et lié au driver.**

```

<html>
  <head>
    <title>PHP Test de connexion avec PDO</title>
  </head>
  <body>
    <h1>Test de connexion avec PDO</h1>
    <?php
      $dsn = "mysql:host=127.0.0.1;port=8889;dbname=FILMS";
      $user = "root";
      $pass = "root";
      $bdd = new PDO ($dsn,$user,$pass);
    ?>
  </body>
</html>

```

## PDO::\_\_construct

(PHP 5 >= 5.1.0, PHP 7, PHP 8, PECL pdo >= 0.1.0)

PDO::\_\_construct — Crée une instance PDO qui représente une connexion à la base

### Description

```

public PDO::__construct(
  string $dsn,
  ?string $username = null,
  ?string $password = null,
  ?array $options = null
)

```

Crée un objet PDO qui représente une connexion à la base.

## MySQL

MySQL can be administered with [phpMyAdmin](#).

To connect to the MySQL server from your own scripts  
use the following connection parameters:

<b>Host</b>	localhost
<b>Port</b>	8889
<b>User</b>	root
<b>Password</b>	root
<b>Socket</b>	/Applications/MAMP/tmp/mysql/mysql.sock

## GESTION DES ERREURS

- La connexion se fait par la construction d'un objet !
- Il peut arriver que la construction de l'objet PDO échoue.
  - Gestion possibles des erreurs
    - Aucune ;)
    - Fin brutale (die)
    - Gérer l'Exception
- En effet si la construction de l'objet ne se fait pas, un objet de type **PDOException** est lancé. A nous de l'attraper ^^
- Remarque : l'objet PDO n'existe donc pas si il y a une erreur !

# GESTION DES ERREURS

- C'est la classe **PDO** qui génère une exception de type **PDOException**
  - Exemple si au lieu de mettre FILMS on mets FILMSS

## Test de connexion avec PDO

**Fatal error:** Uncaught PDOException: SQLSTATE[HY000] [1049] Unknown database 'filmss' in /Applications/MAMP/htdocs/PDO\_Studi\_Part1/pdo\_01.php:12  
Stack trace: #0 /Applications/MAMP/htdocs/PDO\_Studi\_Part1/pdo\_01.php(12):  
PDO->\_\_construct('mysql:host=127....', 'root', 'root') #1 {main} thrown in  
**/Applications/MAMP/htdocs/PDO\_Studi\_Part1/pdo\_01.php** on line **12**

- On a une erreur : **Fatal error !**
- On a un arrêt du programme car on n'a pas récupéré l'exception lancée (**Fatal error: Uncaught PDOException**)
- Pour éviter cela : il faut gérer les exceptions !
- On utilise pour cela le couple **try{} catch{}**.
- On mettra dans try ce qu'on essaye de faire et dans catch ce qu'on doit faire si on a un exception.
- **Attention :** **try{} catch{}** n'est pas là pour gérer les erreurs de programmation mais uniquement gérer au mieux des évènements non prévisibles.  
**Exemples :** la ressources n'est pas là ou n'est plus accessible, la perte de connexion Wifi, panne d'un serveur, on a retiré la clef USB, etc.

# GESTION DES ERREURS

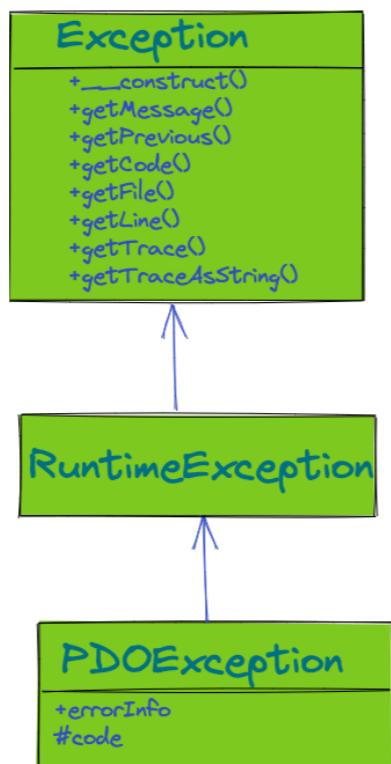
## Test de connexion avec PDO

**Fatal error:** Uncaught PDOException: SQLSTATE[HY000] [1049] Unknown database 'filmss' in /Applications/MAMP/htdocs/PDO\_Studi\_Part1/pdo\_01.php:12  
Stack trace: #0 /Applications/MAMP/htdocs/PDO\_Studi\_Part1/pdo\_01.php(12): PDO->\_\_construct('mysql:host=127....', 'root', 'root') #1 {main} thrown in **/Applications/MAMP/htdocs/PDO\_Studi\_Part1/pdo\_01.php** on line **12**

- **SQLSTATE** est un tableau de cinq caractères. Les deux premiers caractères indiquent la classe générique de la condition, les trois caractères suivants indiquent la sous-classe de la condition générique. Ce code n'est pas spécifique au driver.
  - ➡ HY est une erreur générée par le pilote CLI ou **ODBC**. Ici ODBC
- **[ 1049 ]** erreur spécifique au driver. Erreur pour indiquer que la base n'existe pas.
- On pourra obtenir les mêmes informations lors des requêtes par exemple en utilisant la méthode **errorInfo()**

# GESTION DES ERREURS

- Voilà comment on peut gérer la connexion avec try{} catch{}
- Dès qu'une exception est levée l'execution du code dans le try{} s'arrête (c'est pourquoi coucou n'apparaît pas)
- On va directement dans le catch qui correspond à l'exception levée (On peut avoir plusieurs catch() mais pour notre exemple c'est forcément une exception PDOException).
- Dans le catch{} on utilise les différentes méthodes de la classe PDOException - qui sont en fait des méthodes hérités - pour afficher les messages d'erreur.
- Dans un programme en production il ne faudrait pas les afficher brutes mais les mettre dans des **fichiers de journalisation** (logs) et indiquer un message autre à l'utilisateur et ne pas interrompre le programme.



Le code

```

<html>
  <head>
    <title>PHP Test</title>
  </head>
  <body>
    <?php
      $dsn = "mysql:host=127.0.0.1;port=8889;dbname=FILESS";
      $user = "root";
      $pass = "root";
      try {
        $bdd = new PDO ($dsn,$user,$pass);
        echo "coucou";
      }
      catch(PDOException $e) {
        echo "Le code erreur est : ".$e->getCode();
        echo "<br>Le message : ".$e->getMessage();
        echo "<br>Le fichier : ".$e->getFile();
      }
    ?>
  </body>
</html>
  
```

pdo\_02.php

Le résultat

Le code erreur est : 1049

Le message : SQLSTATE[HY000] [1049] Unknown database 'filmss'

Le fichier : /Applications/MAMP/htdocs/PDO\_Studi\_Part1/pdo\_02.php

# GESTION DES ERREURS AVEC LOG

- L'idée ici va être de gérer les éventuelles erreurs mais en utilisant des logs.
- On a recours à la fonction **error\_log()**
- On va créer deux fichiers
  - Un pour toutes les erreurs
  - Un pour les erreurs de type 1045 c'est à dire mauvaise identification.
  - Ce permettra de mieux voir si on a des tentatives d'attaque style force brute
- La **journalisation** est une recommandation de la **CNIL** voir lien ci-dessous :

<https://www.cnil.fr/fr/la-cnil-publie-une-recommandation-relative-aux-mesures-de-journalisation>

## Le code

```
<html>
  <head>
    <title>PHP Test</title>
  </head>
  <body>
    <?php
      $dsn = "mysql:host=127.0.0.1;port=8889;dbname=FILMSS";
      $user = "root";
      $pass = "roote";
      try {
        $bdd = new PDO ($dsn,$user,$pass);
        echo "coucou";
      }
      catch(PDOException $e) {
        $errorMessage = date('l jS \of F Y h:i:s A ');
        $errorMessage .= " Le code erreur est : ".$e->getCode();
        $errorMessage .= " Le message : ".$e->getMessage();
        $errorMessage .= " Le fichier : ".$e->getFile();
        $errorMessage .= "\n";
        // Chemin du fichier log
        $logFile = "./errors.log";
        // Chemin du fichier accès non autorisé
        $logFileAccess = "./access.log";
        // Enregistrement du message d'erreur dans le fichier log
        error_log($errorMessage, 3, $logFile);
        if ($e->getCode() == 1045)
        {
          error_log($errorMessage, 3, $logFileAccess);
        }
      }
    ?>
  </body>
</html>
```

# REQUÊTE SIMPLE POUR TESTER

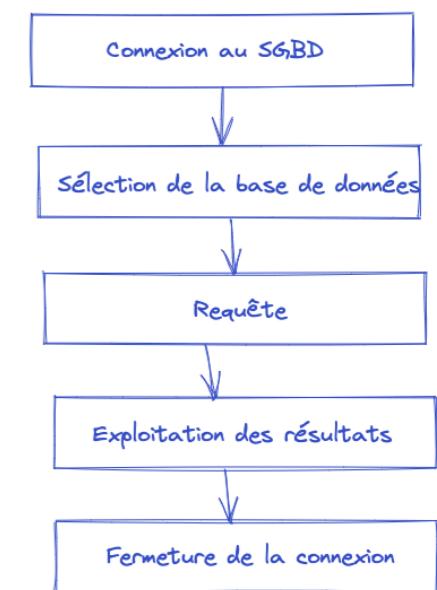
## Code

```

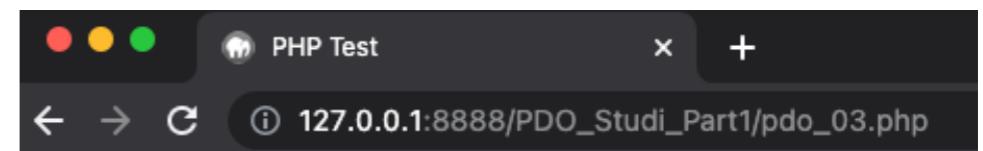
<html>
  <head>
    <title>PHP Test</title>
  </head>
  <body>
    <?php
      $dsn = "mysql:host=127.0.0.1;port=8889;dbname=FILMS";
      $user = "root";
      $pass = "root";
      try {
        $bdd = new PDO ($dsn,$user,$pass);
      }
      catch(PDOException $e) {
        echo "Le code erreur est : ".$e->getCode();
        echo "<br>Le message : ".$e->getMessage();
        echo "<br>Le fichier : ".$e->getFile();
      }
      $req="SELECT nom, prenom FROM acteur";
      foreach($bdd->query($req) as $acteur)
      {
        echo $acteur['nom'].' '.$acteur['prenom']."<br />";
      }
      // Fermeture de la connexion
      $bdd = null;
    ?>
  </body>
</html>

```

pdo\_03.php



## Résultat



Johnny Deep  
Al Pacino  
Suraj Sharma  
Brad Pitt  
Edward Norton  
D'artagnan Edourd

## GESTION DES ERREURS (HORMIS LORS DE LA CONNEXION)

- Ici j'ai fait une erreur dans la requête volontairement en ajoutant un 's' à FROM acteurs
- PDO::ERRMODE\_SILENT ( par défaut )
  - Mode silencieux  
`$bdd->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_SILENT);`
  - Attention : Argument invalide fourni pour foreach() dans /Applications/MAMP/htdocs/PDO\_Studi\_Part1/pdo\_03\_erreur.php à la ligne 20

# GESTION DES ERREURS (HORMIS LORS DE LA CONNEXION)

## • **PDO::ERRMODE\_WARNING**

### • Code de l'erreur et type E\_WARNING

```
$bdd->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING);
```

- Attention : PDO::query(): SQLSTATE[42S02] : Table ou vue de base introuvable : 1146 La table 'films.acteurs' n'existe pas dans /Applications/MAMP/htdocs/PDO\_Studi\_Part1/pdo\_03\_erreur.php à la ligne 21
- Attention : Argument invalide fourni pour foreach() dans /Applications/MAMP/htdocs/PDO\_Studi\_Part1/pdo\_03\_erreur.php à la ligne 21

# GESTION DES ERREURS (HORMIS LORS DE LA CONNEXION)

## • **PDO::ERRMODE\_EXCEPTION**

- Code de l'erreur & lancement de l'exception de type PDOException  
`$bdd->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);`
- Erreur fatale : PDOException non interceptée : SQLSTATE[42S02] : Table ou vue de base introuvable : 1146 La table 'films.acteurs' n'existe pas dans / Applications/MAMP/htdocs/PDO\_Studi\_Part1/pdo\_03\_erreur.php:22 Stack trace : #0 / Applications/MAMP/htdocs/PDO\_Studi\_Part1/ pdo\_03\_erreur.php(22) : PDO->query('SELECT nom, pre...') #1 {main} jeté dans /Applications/MAMP/htdocs/PDO\_Studi\_Part1/pdo\_03\_erreur.php sur ligne 22
- Remarque : On se retrouve dans le cas d'une tentative de création d'un objet de type connexion avec PDO

# RESULTAT D'UN AFFICHAGE AVEC UN TABLEAU ET BOOTSTRAP

## Lister les acteurs

Nom de l'acteur	Prénom de l'acteur
Johnny	Deep
Al	Pacino
Suraj	Sharma
Brad	Pitt
Edward	Norton
D'artagnan	Edourd



MERCI POUR  
VOTRE ATTENTION

