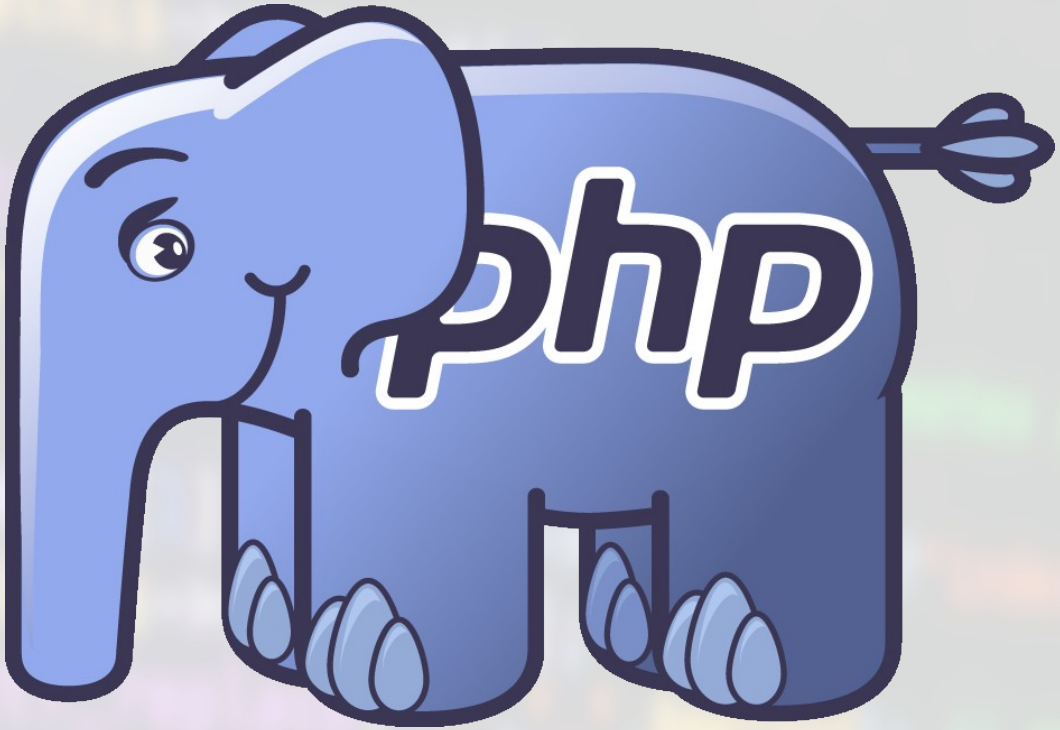
Accés aux Bases de données en PHP (PDO)

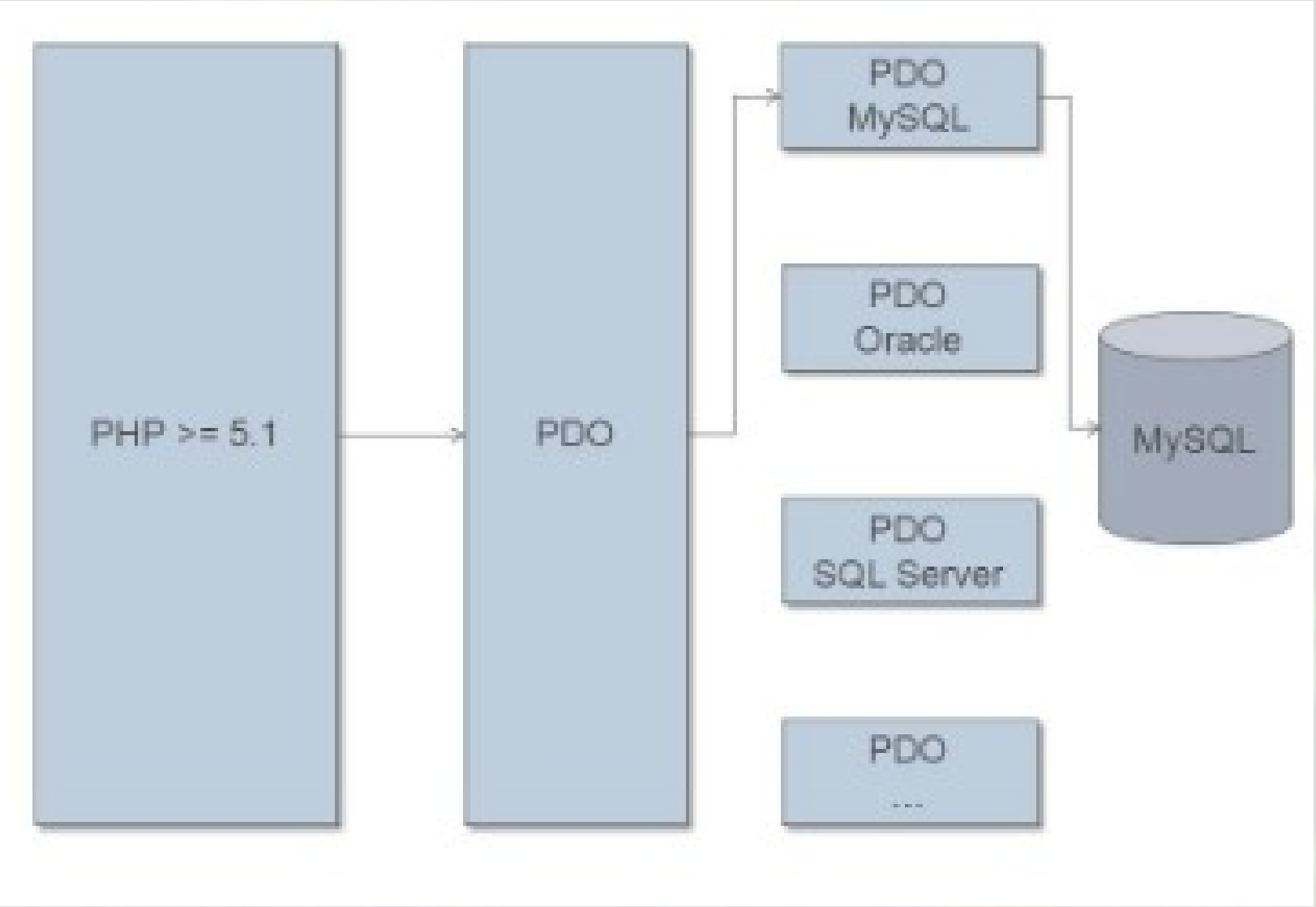


INTRODUCTION

PDO

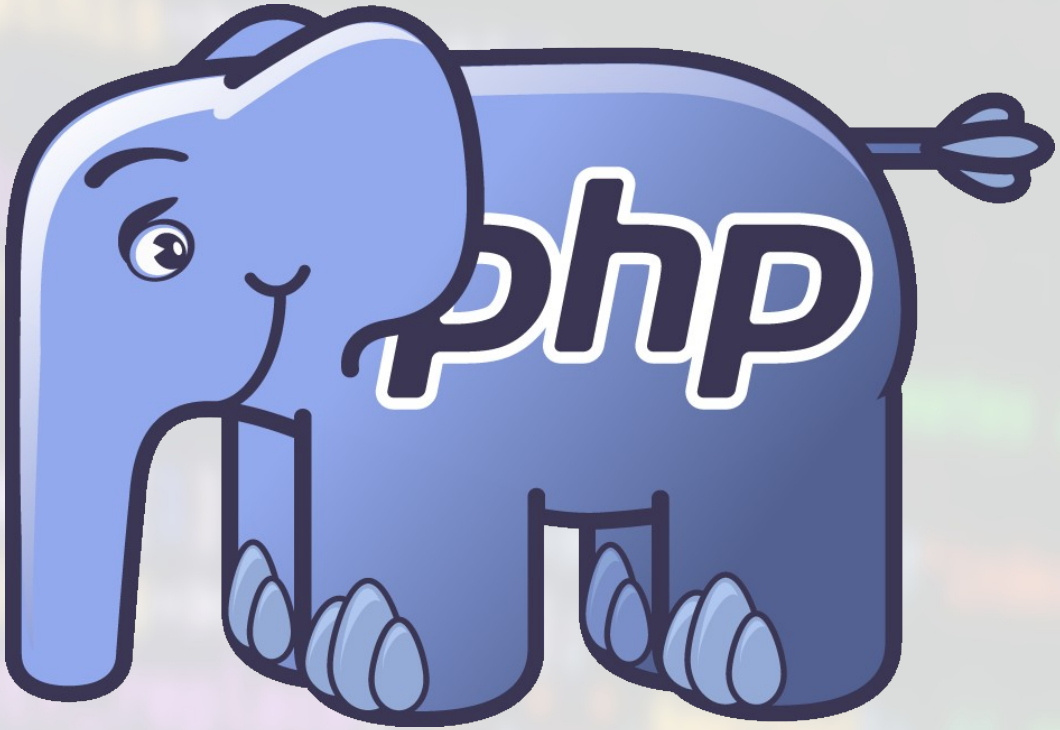
- PDO : **P**HP **D**ata **O**bjects
- Extension PHP fournissant une interface pour accéder à une base de données
- PDO constitue une couche d'abstraction qui intervient entre l'application PHP et un système de gestion de base de données (SGDB)
- PDO facilite donc la **migration** vers un autre SGBD puisqu'il n'est plus nécessaire de changer le code déjà développé.
- Interface orientée objet

Architecture PDO



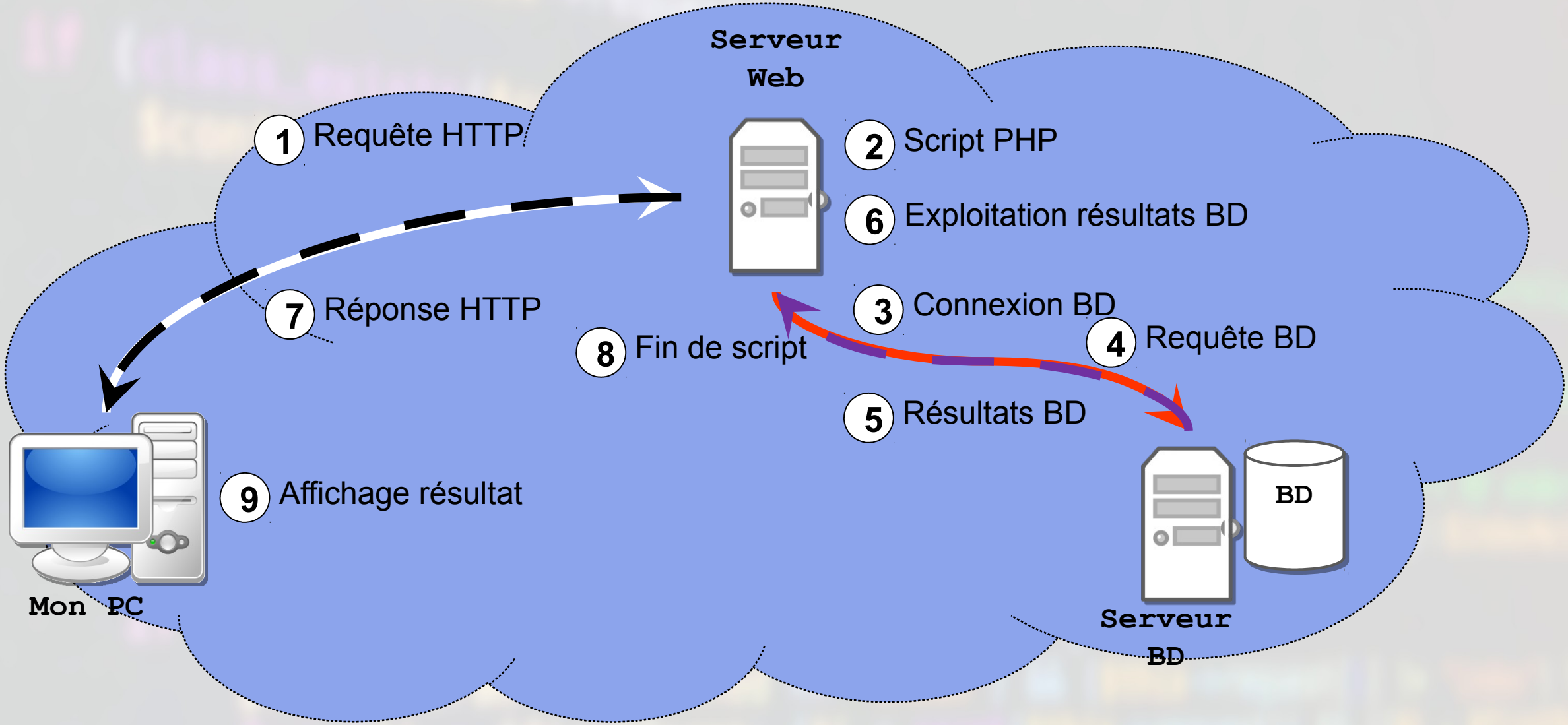
Bases de données supportées

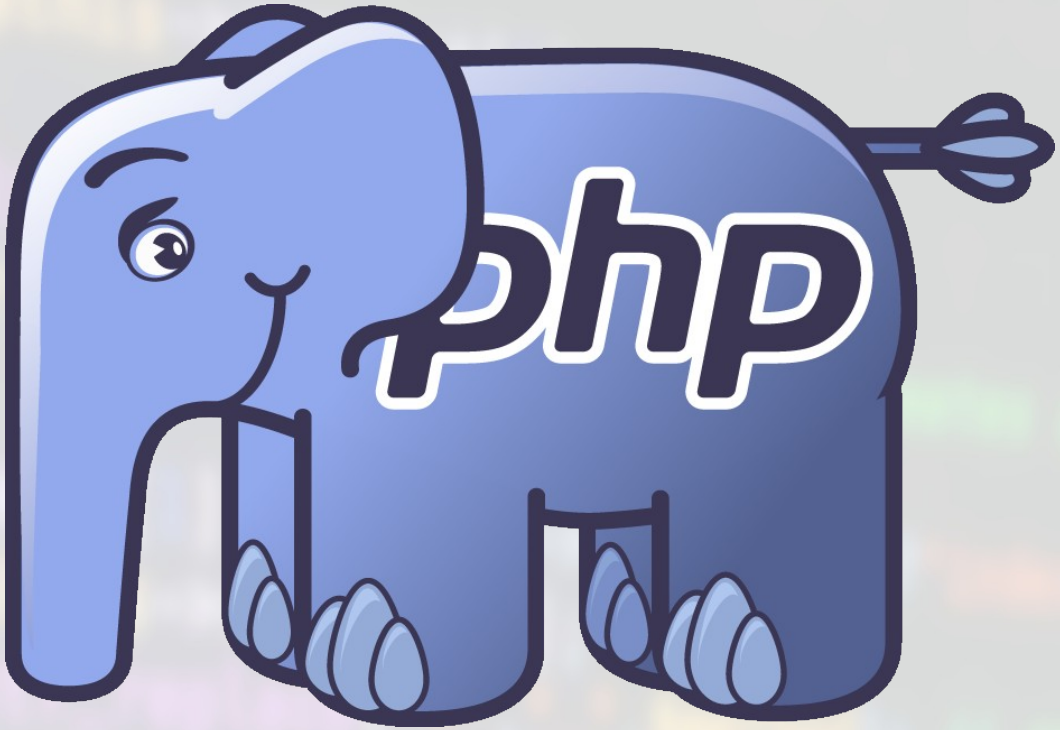
Nom du pilote	Bases de données prises en charge
PDO_CUBRID	Cubrid
PDO_DBLIB	FreeTDS / Microsoft SQL Server / Sybase
PDO_FIREBIRD	Firebird
PDO_IBM	IBM DB2
PDO_INFORMIX	IBM Informix Dynamic Server
PDO_MYSQL	MySQL 3.x/4.x/5.x
PDO_OCI	*Oracle Call Interface*
PDO_ODBC	ODBC v3 (IBM DB2, unixODBC et win32 ODBC)
PDO_PGSQL	PostgreSQL
PDO_SQLITE	SQLite 3 et SQLite 2
PDO_SQLSRV	Microsoft SQL Server / SQL Azure
PDO_4D	4D



ÉCHANGES DE DONNÉES

Échanges de données





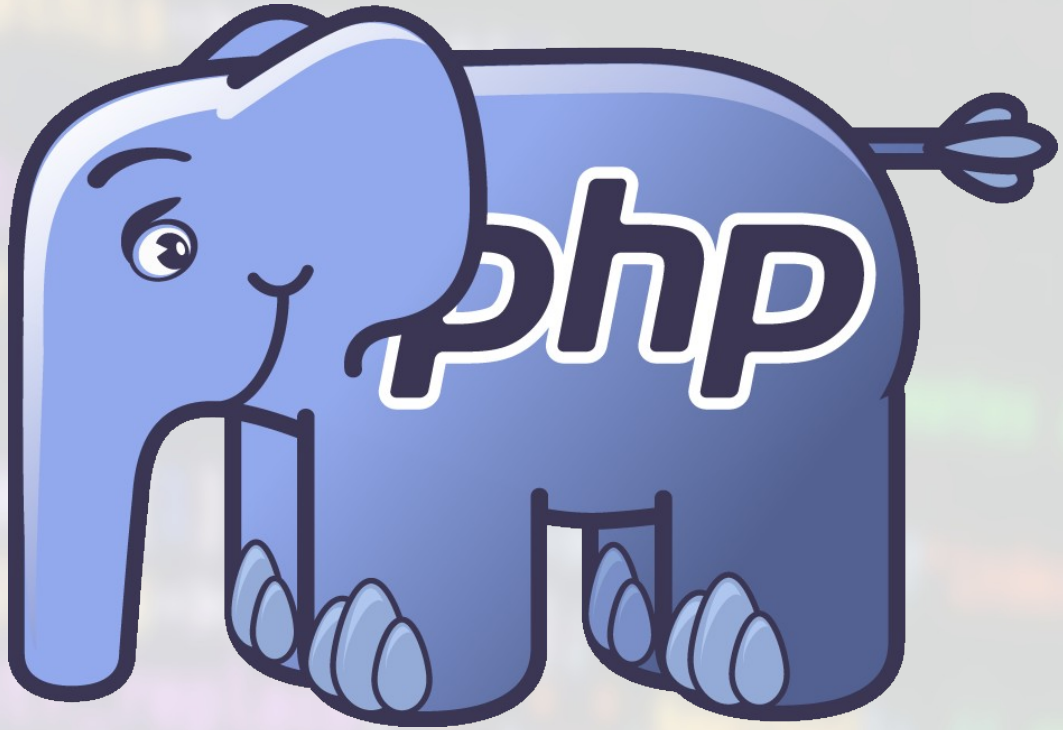
LES CLASSES DE PDO

Classes prédéfinies

- **PDO** : connexion PHP / base de données
  - `__construct()`
  - `exec(), prepare(), query()`
  - `errorCode(), errorInfo()`
  - `getAttributes(), setAttribute()`
  - `lastInsertId(), quote()`
  - `beginTransaction()`
  - `commit(), rollBack()`
  - `getAvailableDrivers()`

Classes prédéfinies

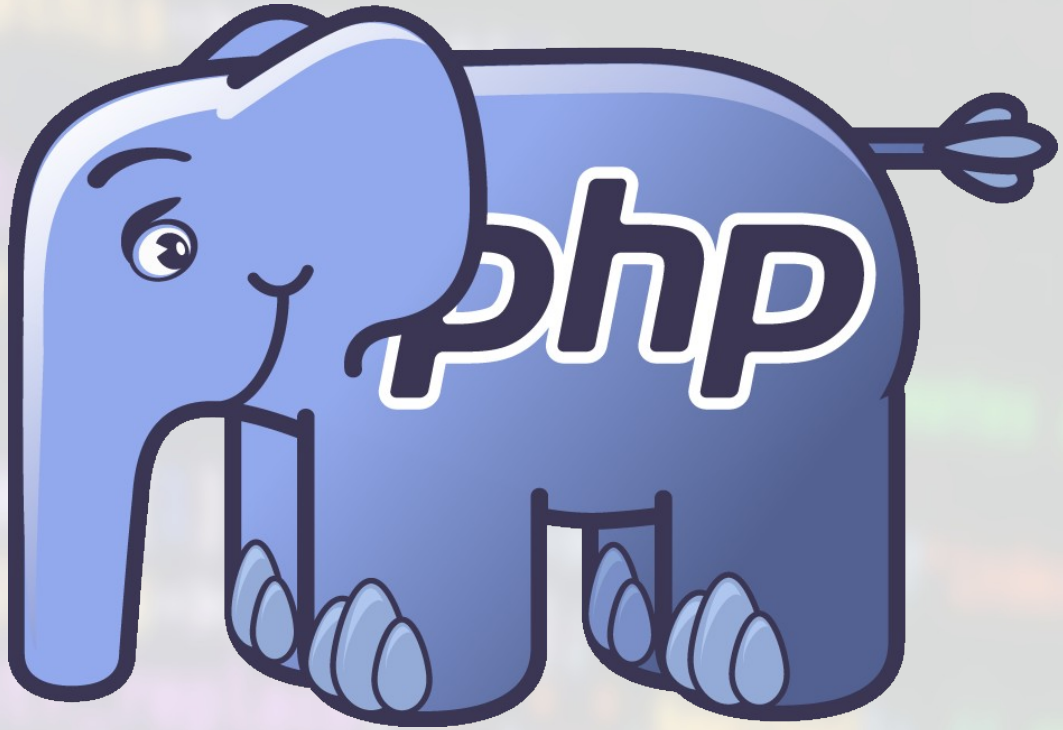
- **PDOStatement** : requête préparée, jeu de résultats
  - `bindColumn()`, `bindParam()`, `bindValue()`, `closeCursor()`
  - `errorCode()`, `errorInfo()`
  - `fetch()`, `fetchAll()`, `fetchColumn()`, `fetchObject()`, `setFetchMode()`, `nextRowset()`
  - `rowCount()`, `columnCount()`, `getColumnMeta()`
  - `getAttribute()`, `setAttribute()`
  - `execute()`
  - `debugDumpParams()`



COMMENT SE CONNECTER ?

Connexions et gestionnaire de connexion

- Instanciation d'un objet **PDO**
- `$dbh=new PDO(DSN [, user [, pass [, options]]]);`
- *DSN* : **D**ata **S**ource **N**ame
  - `nom_du_driver:syntaxe_spécifique_au_driver`
  - Ex : `'mysql:host=localhost;dbname=ma_base'`
- *user* : nom d'utilisateur, *pass* : mot de passe
- *options* : tableau associatif
  - spécifiques au driver
  - Ex : `array(PDO::ATTR_PERSISTENT => true));`
- Fin de connexion : `$dbh=null; /* ou */ unset($dbh);`



GESTION DES ERREURS

Gestion des erreurs de connexion

- Connexion par construction d'un objet
- Gestion envisageable des erreurs
  - Aucune
  - Fin brutale (**`exit`**, **`die`**)
  - État
  - Exception
- En cas d'erreur de connexion
  - Objet **`PDOException`** lancé
  - **`PDOException`** hérite de **`Exception`**

Gestion des erreurs de connexion

```php
<?php
try {
 $dbh = new PDO('mysql:host=dbs;dbname=music',
                $user, $pass) ;
 …
 $dbh = null ;
}
catch (PDOException $e) {
 echo "<p>Erreur: ".$e->getMessage() ;
 die() ;
}
```

Gestion des erreurs (hormis connexion)

- `PDO::ERRMODE_SILENT` (*par défaut*)
  - Mode silencieux, mise en place d'un code d'erreur
  - `PDO` : `errorCode()` / `errorInfo()`
  - `PDOStatement` : `errorCode()` / `errorInfo()`
- `PDO::ERRMODE_WARNING`
  - Mise en place du code d'erreur
  - Émission d'une erreur de type `E_WARNING`
- `PDO::ERRMODE_EXCEPTION`
  - Mise en place du code d'erreur
  - Objet `PDOException` lancé

Gestion des erreurs (hormis connexion)

```php
<?php
try {
 $dbh = new PDO('mysql:host=dbs;dbname=music',
                $user, $pass) ;
 $dbh->setAttribute(PDO::ATTR_ERRMODE,
                    PDO::ERRMODE_EXCEPTION);
 …
 $dbh = null ;
}
catch (PDOException $e) {
 echo "<p>Erreur: ".$e->getMessage() ;
 die() ;
}
```

Gestion des erreurs : code d'erreur

```php
<?php
 $pdo = new PDO("mysql:host=dbs;dbname=music") ;
 $pdostat = $pdo->query("COUCOU") ;
 if ($pdo->errorCode()) {
  echo "ERREUR !!\n" ;
  echo "<pre>\n" ;
  var_dump($pdo->errorInfo()) ;
  echo "</pre>\n" ;
 }
```

```
ERREUR !!
array(3) {
  [0]=> string(5) "42000"
  [1]=> int(1064)
  [2]=> string(47) "Erreur de syntaxe près de 'COUCOU' à la ligne 1"
}
```

Code SQLSTATE

Code erreur spécifique du driver

Chaîne erreur spécifique au driver

Gestion des erreurs : exceptions

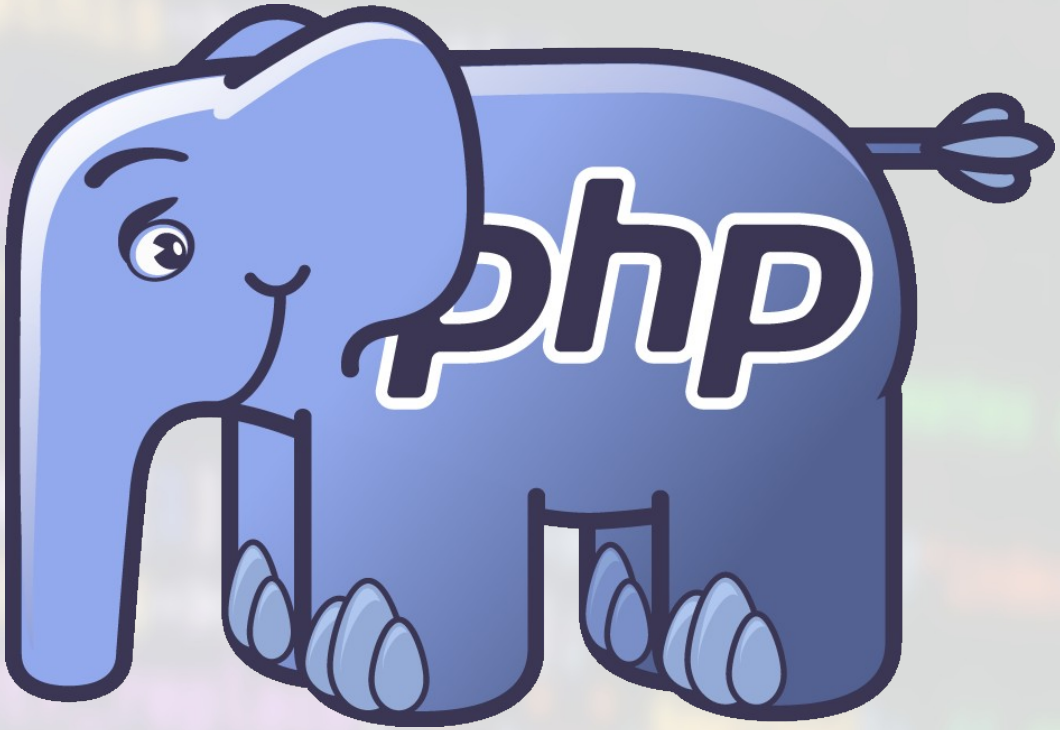
```php
<?php
 try {
  $pdo = new PDO("mysql:host=dbs;dbname=music") ;
  $pdo->setAttribute(PDO::ATTR_ERRMODE,
                     PDO::ERRMODE_EXCEPTION) ;
  $pdostat = $pdo->query("COUCOU") ;
 }
 catch (Exception $e) {
  echo "<p>ERREUR : ".$e->getMessage() ;
 }
```

Code erreur spécifique du driver

Code SQLSTATE

Chaîne erreur spécifique au driver

```
ERREUR : SQLSTATE[42000]: Syntax error or access violation: 1064
 Erreur de syntaxe près de 'COUCOU' à la ligne 1
```



EFFECTUER UNE REQUÊTE

Exécution d'une requête

- **PDO::query (string *statement*): PDOStatement**


```php
<?php
 try {
  $pdo = new PDO("mysql:host=dbs;dbname=music") ;
  $pdostat = $pdo->query("SELECT * FROM artist") ;
 }
 catch (Exception $e) {
  echo "<p>ERREUR : ".$e->getMessage() ;
 }
```

Exploitation des résultats d'une requête

- Récupération des données ligne à ligne
- Une ligne peut être :
  - un tableau indexé
  - un tableau associatif
  - un tableau mixte (par défaut)
  - un objet anonyme
  - un objet d'une classe définie par l'utilisateur
- Récupération des données d'une colonne

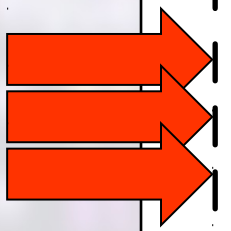
Parcours du résultat d'une requête

- Parcourir le résultat de la requête

```
SELECT *
FROM morceau
ORDER BY mor_id
```

Résultat de requête

```
+--------+-------------------------------+
| mor_id | mor_nom                       |
+--------+-------------------------------+
|    872 | With A Little Help From My Friends |
|    873 | The Letter                    |
|    874 | Marjorine                     |
|    875 | Midnight Rider                |
|    876 | You Are So Beautiful          |
|    877 | Feelin' Allright              |
|    878 | Cry Me A River                |
...
```



Curseur interne

Mode de récupération des résultats d'une requête

```php
$pdostat = $pdo->query(
    "SELECT id, name FROM artist");
$r = $pdostat->fetch(fetch_mode));
```

fetch_mode

PDO::FETCH_NUM

```php
// array
$r[0]
$r[1]
```

PDO::FETCH_CLASS, *ma_classe*

```php
// ma_classe
$r->id
$r->name
```

PDO::FETCH_OBJ

```php
// object
$r->id
$r->name
```

PDO::FETCH_ASSOC

```php
// array
$r['id']
$r['name']
```

Exploitation des résultats d'une requête (1)

```
try {
 $pdo=new PDO("mysql:host=dbs;dbname=music");
 $pdo->setAttribute(PDO::ATTR_ERRMODE,
                    PDO::ERRMODE_EXCEPTION);
 $pdostat = $pdo->query("SELECT id, name FROM artist");
 $pdostat->setFetchMode(PDO::FETCH_ASSOC);
 foreach ($pdostat as $ligne) {
     echo "<p>" . $ligne['name'] . "\n";
 }
}
catch (Exception $e) {
  echo "<p>ERREUR : ".$e->getMessage();
}
```

Exploitation des résultats d'une requête (2)

```
try {
 $pdo=new PDO("mysql:host=dbs;dbname=music");
 $pdo->setAttribute(PDO::ATTR_ERRMODE,
                    PDO::ERRMODE_EXCEPTION);
 $pdostat = $pdo->query("SELECT id, name FROM artist");
 foreach ($pdostat->fetchAll(PDO::FETCH_ASSOC) as $ligne) {
     echo "<p>" . $ligne['name'] . "\n";
 }
}
catch (Exception $e) {
  echo "<p>ERREUR : ".$e->getMessage();
}
```

Exploitation des résultats d'une requête (3)

```
try {
 $pdo=new PDO("mysql:host=localhost;dbname=mysql") ;
 $pdo->setAttribute(PDO::ATTR_ERRMODE,
                    PDO::ERRMODE_EXCEPTION);
 $pdostat = $pdo->query("SELECT name FROM user") ;
 while (($ligne = $pdostat->fetch(PDO::FETCH_ASSOC)) !== false) {
     echo "<p>" . $ligne['name'] . "\n" ;
 }
}
catch (Exception $e) {
   echo "<p>ERREUR : ".$e->getMessage() ;
}
```

Modes de récupération des données (1)

- **`PDO::FETCH_ASSOC`**
  - retourner chaque ligne dans un tableau indexé par les noms des colonnes comme elles sont retournées dans le jeu de résultats correspondant. Si le jeu de résultats contient de multiples colonnes avec le même nom, **`PDO::FETCH_ASSOC`** retourne une seule valeur par nom de colonne.
- **`PDO::FETCH_NUM`**
  - retourner chaque ligne dans un tableau indexé par le numéro des colonnes comme elles sont retournées dans le jeu de résultats correspondant, en commençant à 0.

Modes de récupération des données (2)

- **`PDO::FETCH_BOTH`** (*par défaut*)
  - retourner chaque ligne dans un tableau indexé par les noms des colonnes ainsi que leurs numéros, comme elles sont retournées dans le jeu de résultats correspondant, en commençant à 0.
- **`PDO::FETCH_OBJ`**
  - retourner chaque ligne dans un objet avec les noms de propriétés correspondant aux noms des colonnes comme elles sont retournées dans le jeu de résultats.

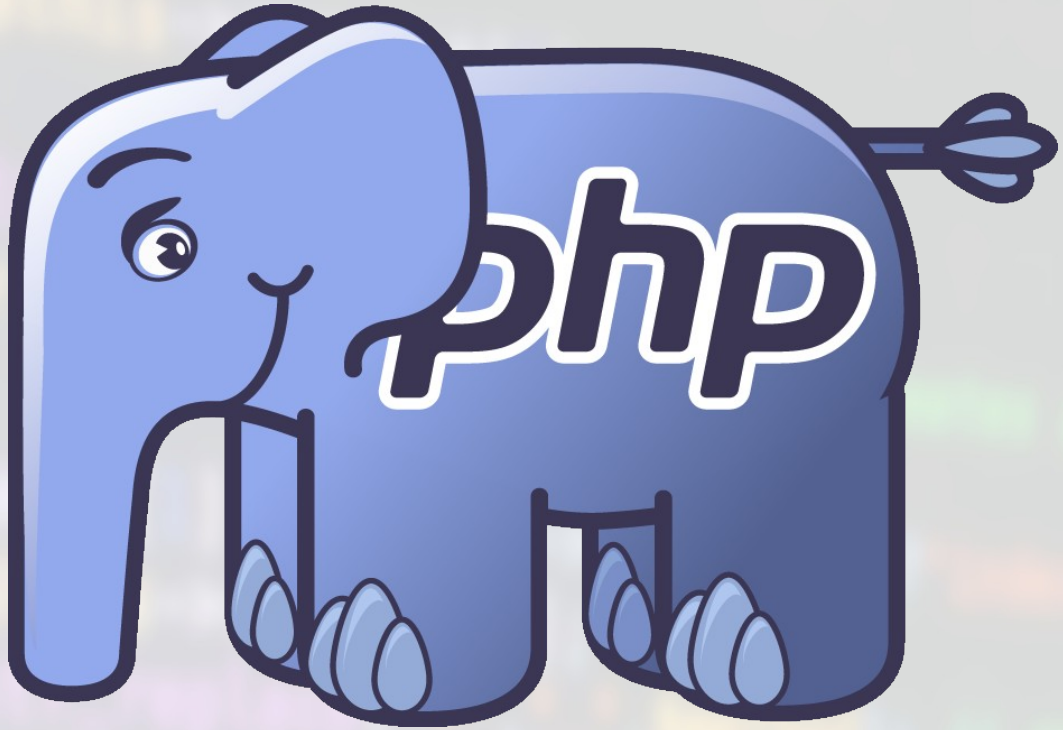
Modes de récupération des données (3)

- **`PDO::FETCH_BOUND`**
  - retourner **`true`** et assigner les valeurs des colonnes du jeu de résultats dans les variables PHP auxquelles elles sont liées avec la méthode **`PDOStatement::bindParam()`** ou la méthode **`PDOStatement::bindColumn()`**.
- **`PDO::FETCH_CLASS | PDO::FETCH_CLASSTYPE`**
  - retourner une nouvelle instance de la classe demandée, liant les colonnes aux propriétés nommées dans la classe.
    Nom de la classe = 1ère colonne.

Exemple avec `PDO::FETCH_CLASS`

```php
$stmt = $pdo->query(<<<SQL
    SELECT id, name
    FROM artist
    WHERE id = 12
SQL
) ;
$stmt->setFetchMode(PDO::FETCH_CLASS, 'Artist') ;
if (($object = $stmt->fetch()) !== false) {
    return $object ;
}
```

Instancie un objet de la classe `Artist`
dont les attributs sont supposés être `id` et `name`,
exactement le nom des champs de la table



REQUÊTES PRÉPARÉES

Préparation d'une requête

- Déroulement d'une requête SQL
  1. Analyse
  2. Compilation
  3. Optimisation
  4. Exécution
- Exécution répétée d'une requête : 1+2+3+4
- Préparation d'une requête : 1+2+3
- Exécution répétée d'une requête préparée : 4
- Préparation en fonction de paramètres :
  - Anonymes
  - Nommés

Préparation d'une requête

- `PDO::prepare(string statement [,array driver_options]): PDOStatement`
  - *`statement`* : la requête à préparer.
    Peut contenir des paramètres anonymes (`?`) ou nommés (`:nom`)
  - *`driver_options`* : tableau d'options du driver
  - retourne un objet `PDOStatement` qui effectuera l'association des paramètres et exécutera la requête

```
$pdo = new PDO("mysql:host=localhost;dbname=mysql");
$pdostat = $pdo->prepare("SELECT id, name
                          FROM artist
                          WHERE id= ?");
```

Association des paramètres d'une requête

- `PDOStatement::bindValue(`*`mixed parameter`*`, `*`mixed value`*` [, `*`int data_type`*`]): bool`
  - *`parameter`* : le paramètre (nom ou position [1…n])
  - *`value`* : sa valeur
  - *`data_type`* : le type de la valeur
    - **`PDO::PARAM_BOOL`** booléen.
    - **`PDO::PARAM_NULL`** NULL SQL.
    - **`PDO::PARAM_INT`** INTEGER SQL.
    - **`PDO::PARAM_STR`** CHAR, VARCHAR ou autre chaîne.
    - **`PDO::PARAM_LOB`** "objet large" SQL.
- `PDOStatement::execute([`*`array parameters`*`]): bool`
  - *`parameters`* : tableau associatif ou indexé des valeurs

Préparation puis exécution d'une requête (1)

```php
$pdo = new PDO("mysql:host=dbs;dbname=app") ;
$pdo->setAttribute(PDO::ATTR_ERRMODE,
                   PDO::ERRMODE_EXCEPTION);
$pdostat = $pdo->prepare("SELECT * FROM user
                          WHERE login = ?") ;
$pdostat->bindValue(1, 'root') ;
$pdostat->execute() ;
// Utilisation du résultat
$pdostat->bindValue(1, 'cutrona') ;
$pdostat->execute() ;
// Utilisation du résultat
```

paramètre anonyme

Exécution de la requête

Préparation puis exécution d'une requête (2)

```php
$pdo = new PDO("mysql:host=dbs;dbname=app") ;
$pdo->setAttribute(PDO::ATTR_ERRMODE,
                   PDO::ERRMODE_EXCEPTION);
$pdostat = $pdo->prepare("SELECT * FROM user
                          WHERE login = :utilisateur") ;
$pdostat->bindValue(':utilisateur', 'root') ;
$pdostat->execute() ;
// Utilisation du résultat
$pdostat->bindValue(':utilisateur', 'cutrona') ;
$pdostat->execute() ;
// Utilisation du résultat
```

paramètre nommé

Exécution de la requête

Préparation puis exécution d'une requête (3)

```php
$pdo = new PDO("mysql:host=dbs;dbname=app") ;
$pdo->setAttribute(PDO::ATTR_ERRMODE,
                   PDO::ERRMODE_EXCEPTION);
$pdostat = $pdo->prepare("SELECT * FROM user
                          WHERE login = ?") ;
$pdostat->execute(array('root')) ;
// Utilisation du résultat
$pdostat->execute(array('cutrona')) ;
// Utilisation du résultat
```

paramètre anonyme

Exécution de la requête

Préparation puis exécution d'une requête (4)

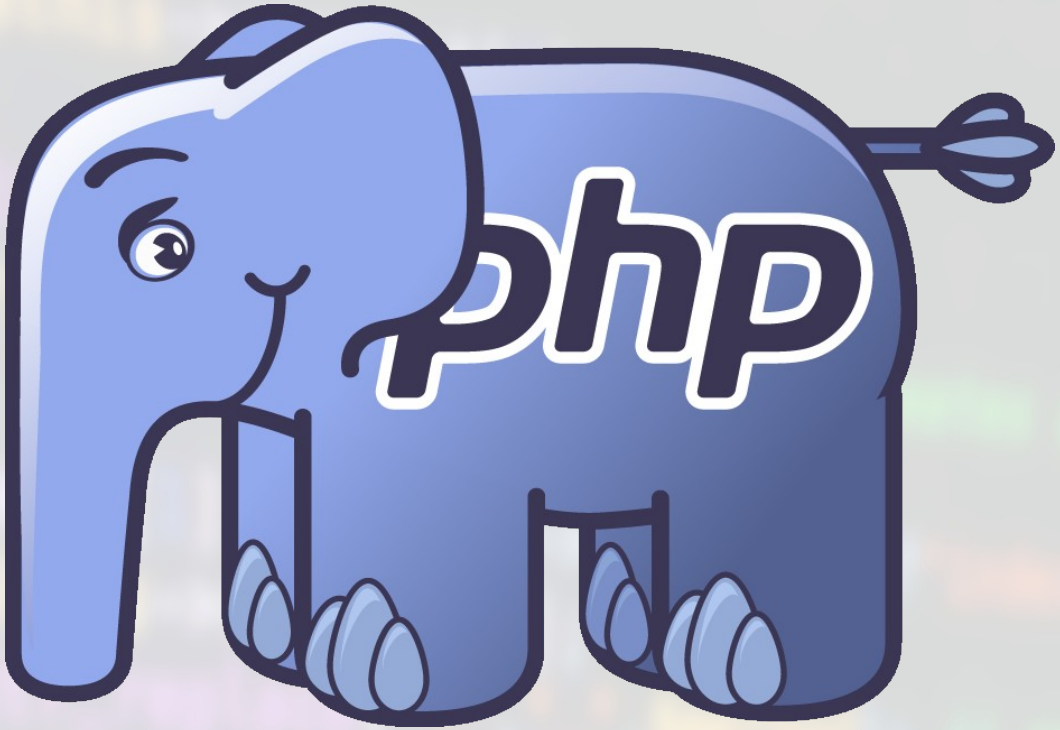
```php
$pdo = new PDO("mysql:host=dbs;dbname=app") ;
$pdo->setAttribute(PDO::ATTR_ERRMODE,
                   PDO::ERRMODE_EXCEPTION);
$pdostat = $pdo->prepare("SELECT * FROM user
                          WHERE login = :utilisateur") ;
$pdostat->execute(
                array(':utilisateur' => 'root')) ;
// Utilisation du résultat
$pdostat->execute(
                array(':utilisateur' => 'cutrona')) ;
// Utilisation du résultat
```

paramètre nommé

Exécution de la requête

Intérêt des requêtes préparées

- Amélioration des performances en cas d'exécutions répétées
- Émulation faite par `PDO` si le driver ne les supporte pas nativement
- Protection automatique des valeurs des paramètres pour interdire les attaques par injection de code SQL



ATTAQUE PAR INJECTION SQL

Attaque par injection SQL ?

- Exemple : validation d'un login/password sur un site
- Requête consistant à trouver un enregistrement correspondant au couple login/password fourni par l'utilisateur
- Requête naïve :

```
SELECT *
FROM membre
WHERE login='{$_POST['login']}'
   AND passwd='{$_POST['passwd']}'
```

- Et si on essayait de fournir un mot de passe un peu particulier…

Exemple concret d'injection SQL (1)

```php
$pdo = new PDO("mysql:host=dbs;dbname=app") ;
$pdostat = $pdo->query($req = <<<SQL
    SELECT *
    FROM membre
    WHERE login  = '{$_POST['login']}'
      AND passwd = '{$_POST['passwd']}'
SQL
    ) ;
echo "Requête:\n$req\n" ;
if (($utilisateur = $pdostat->fetch())) !== false)
  echo "Bienvenue {$utilisateur['nom']}" ;
else
  echo "Désolé..." ;
```

Exemple concret d'injection SQL (2)

Saisie de l'utilisateur par formulaire :

- login : `whatever`
- pass : `who_cares?`

```
Requête:
    SELECT *
    FROM membre
    WHERE login='whatever'
      AND passwd='who_cares?'
 Désolé...
```

Exemple concret d'injection SQL (3)

Saisie de l'utilisateur :

- login : `whatever`
- pass : `who_cares?' OR tru`

Requête:

```
SELEC
FR
      ogin='whatever'
   passwd='who_cares?' OR true!=''
enue John
```

Donne toutes les lignes de la table membre
Le premier membre est certainement l'administrateur !

Protection contre les injections SQL (1)

```
$pdo = new PDO("mysql:host=dbs;dbname=app") ;
$pdostat = $pdo->prepare($req = <<<SQL
     SELECT *
     FROM membre
     WHERE login  = ?
       AND passwd = ?
SQL
 ) ;
$pdostat->execute(array($_POST['login'],
                        $_POST['passwd'])) ;
if (($utilisateur = $pdostat->fetch()) !== false)
     { echo "Bienvenue {$utilisateur['nom']}\n" ; }
else { echo "Désole...\n" ; }
```

Protection contre les injections SQL (2)

```
$pdo = new PDO("mysql:host=dbs;dbname=app") ;
$login  = $pdo->quote($_POST['login']) ;
$passwd = $pdo->quote($_POST['passwd']) ;
$pdostat = $pdo->query($req = <<<SQL
    SELECT *
    FROM membre
    WHERE login  = $login
      AND passwd = $p
SQL
 ) ;
echo "Requête:\n$req
if (($utilisateur =
     { echo "Bienven
else { echo "Désole.
```

```
Requête:
    SELECT *
    FROM membre
    WHERE login='whatever'
      AND passwd='who_cares?\' OR true!=\''
Désolé...
```

Protection contre les injections SQL : conclusion

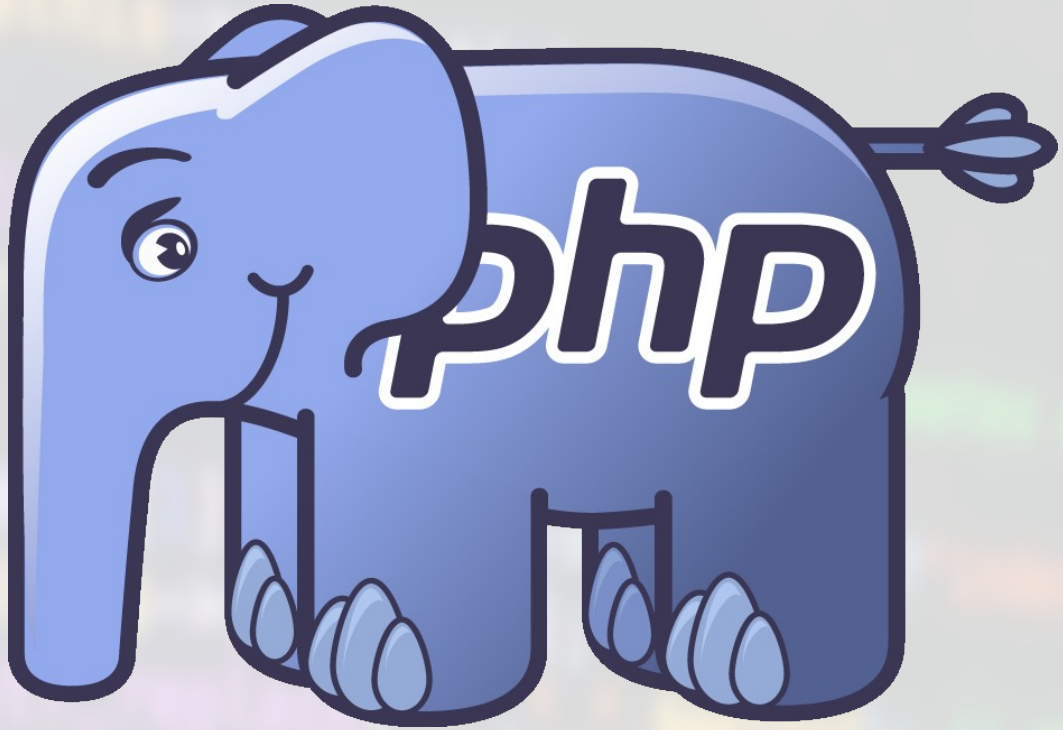
Jamais de substitution de variable dans une requête SQL

Jamais de concaténation dans une requête SQL

Préparer une requête = augmenter son efficacité (cache)

Préparer une requête = pouvoir la paramétrer

Paramétrer une requête = se protéger contre l'injection de code



GESTION DES TRANSACTIONS

Transactions

- Transactions :
  - Atomicité, Consistance, Isolation et Durabilité
  - BEGIN puis COMMIT ou ROLLBACK
- Mode PDO par défaut :
  - Chaque requête est validée automatiquement
- **`PDO::beginTransaction()`**
- **`PDO::commit()`**
- **`PDO::rollBack()`**
- Tous les moteurs ne supportent pas les transactions
  Ⓟ **`PDOException`**