# PDO (PHP Data Objects) — Synthèse rapide

## 1 C'est quoi PDO ?

**PDO** (*PHP Data Objects*) est une couche d'abstraction qui permet de communiquer avec différentes bases de données *via* les mêmes fonctions PHP. L'API reste identique, que l'on utilise MySQL, PostgreSQL, SQLite, etc.

## 2 À quoi ça sert ?

### 1. Unifier l'accès aux bases de données

Avec PDO, le code change seulement au niveau de la chaîne DSN, le reste de l'API est *identique :*

```php
// MySQL
$pdo = new PDO("mysql:host=localhost;dbname=test", $user, $pass);

// PostgreSQL
$pdo = new PDO("pgsql:host=localhost;dbname=test", $user, $pass);

// SQLite
$pdo = new PDO("sqlite:/chemin/vers/database.sqlite");
```

### 2. Sécurité renforcée contre les injections SQL

Les *requêtes préparées* avec `bindValue()` sont plus sécurisées avec PDO :

```php
// AVEC PDO et bindValue() (s curis )
$stmt = $pdo->prepare("SELECT * FROM users WHERE email = :email AND
    password = :password");
$stmt->bindValue(':email', $email, PDO::PARAM_STR);
$stmt->bindValue(':password', $password, PDO::PARAM_STR);
$stmt->execute();

// SANS PDO (dangereux - injection SQL possible)
$sql = "SELECT * FROM users WHERE email = '$email' AND password = '
    $password'";
$result = $mysqli->query($sql);
```

### 3. Gestion d'erreurs améliorée

PDO s'appuie sur les *exceptions*, plus modernes et structurées :

```php
// PDO - gestion par exceptions
try {
    $pdo = new PDO(...);
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
```

```
    echo "Erreur : " . $e->getMessage();
}

// MySQLi - style plus ancien
if ($mysqli->connect_error) {
    die("Erreur : " . $mysqli->connect_error);
}
```

# 3 PDO vs MySQLi : comparaison pratique

### INSERT avec MySQLi

```
$stmt = $mysqli->prepare("INSERT INTO animaux (nom, espece, age) VALUES (?, ?, ?)");
$stmt->bind_param("ssi", $nom, $espece, $age); // "ssi" = string, string, integer
$stmt->execute();
```

### INSERT avec PDO et bindValue() (Recommandé)

```
$stmt = $pdo->prepare(
  "INSERT INTO animaux (nom, espece, age) VALUES (:nom, :espece, :age)"
);
// bindValue() avec types explicites - plus s curis
$stmt->bindValue(':nom', $nom, PDO::PARAM_STR);
$stmt->bindValue(':espece', $espece, PDO::PARAM_STR);
$stmt->bindValue(':age', $age, PDO::PARAM_INT);
$stmt->execute();
```

### INSERT avec PDO (paramètres positionnels et bindValue)

```
$stmt = $pdo->prepare("INSERT INTO animaux (nom, espece, age) VALUES (?, ?, ?)");
// bindValue() avec param tres positionnels
$stmt->bindValue(1, $nom, PDO::PARAM_STR);
$stmt->bindValue(2, $espece, PDO::PARAM_STR);
$stmt->bindValue(3, $age, PDO::PARAM_INT);
$stmt->execute();
```

### SELECT avec PDO et bindValue()

```
function get_animal_by_id(int $id) {
    $pdo = db();
    $stmt = $pdo->prepare(
        "SELECT id, nom, espece, age FROM animaux WHERE id = :id"
    );
    $stmt->bindValue(':id', $id, PDO::PARAM_INT);
    $stmt->execute();
    return $stmt->fetch() ?: null;
}
```

## 4 Les différentes méthodes de binding PDO

### bindValue() (Recommandé)

```php
// Lie une valeur avec son type
$stmt->bindValue(':nom', $nom, PDO::PARAM_STR);
$stmt->bindValue(':age', $age, PDO::PARAM_INT);
$stmt->bindValue(':actif', $actif, PDO::PARAM_BOOL);
$stmt->bindValue(':option', null, PDO::PARAM_NULL);
$stmt->execute();
```

### bindParam() (Pour références)

```php
// Lie une variable par r f rence (change si la variable change)
$stmt->bindParam(':nom', $nom, PDO::PARAM_STR);
$stmt->bindParam(':age', $age, PDO::PARAM_INT);
$stmt->execute();
```

### execute() avec tableau (Plus court mais moins explicite)

```php
// M thode rapide mais sans sp cification de type
$stmt->execute([
    ':nom' => $nom,
    ':espece' => $espece,
    ':age' => $age
]);
```

## 5 Quand utiliser MySQLi ?

— Projets très simples et spécifiques à MySQL.
— Besoin de performances maximales *dans un contexte MySQL uniquement.*
— Code *legacy* à maintenir qui est déjà en MySQLi.

## 6 Meilleures pratiques PDO

— Utilisez toujours `bindValue()` avec le type approprié.
— Activez le mode exception : `setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION)`.
— Utilisez `utf8mb4` pour le charset.
— Fermez les connexions en mettant `$pdo = null`.

**Remarque sécurité :** pour les mots de passe, utilisez toujours `password_hash()` et `password_verify()` côté PHP, quel que soit l'API choisie (PDO ou MySQLi).