

# PDO 3/3

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\\_Part2](https://github.com/SebInfo/PDO_Studi_Part2)

# PRINCIPE DES REQUETES PREPAREES



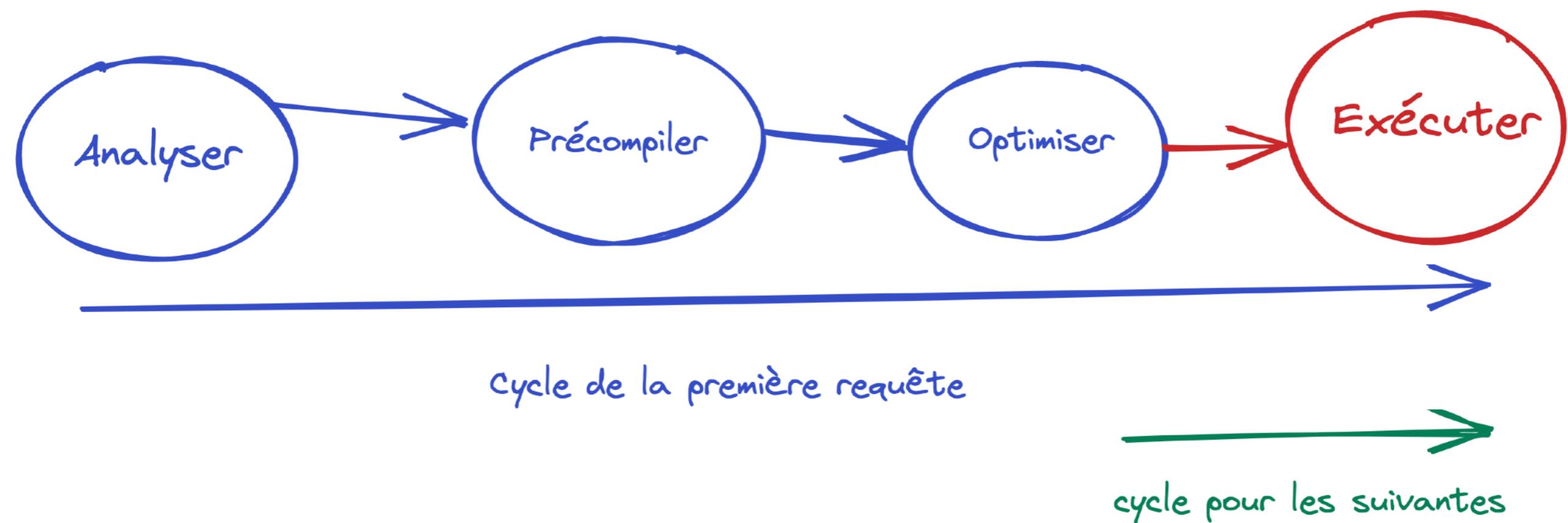
# LES REQUÊTES PRÉPARÉES : PRINCIPES

---

- ▶ Préparer une requête c'est **créer un modèle de requête** qui est enregistré sur le SGBD le temps de l'execution du script.  
(contrairement au procédures stockées qui le sont de manière permanente)
- ▶ Quand on a besoin on peut alors faire appel à ce modèle et ceux de manière plus rapide. On peut donc **appeler plusieurs fois une requête préparé avec des valeurs différentes.**
- ▶ La base de données MySQL supporte les requêtes préparées ou encore requêtes paramétrables.

# LES REQUÊTES PRÉPARÉES LA RAPIDITE

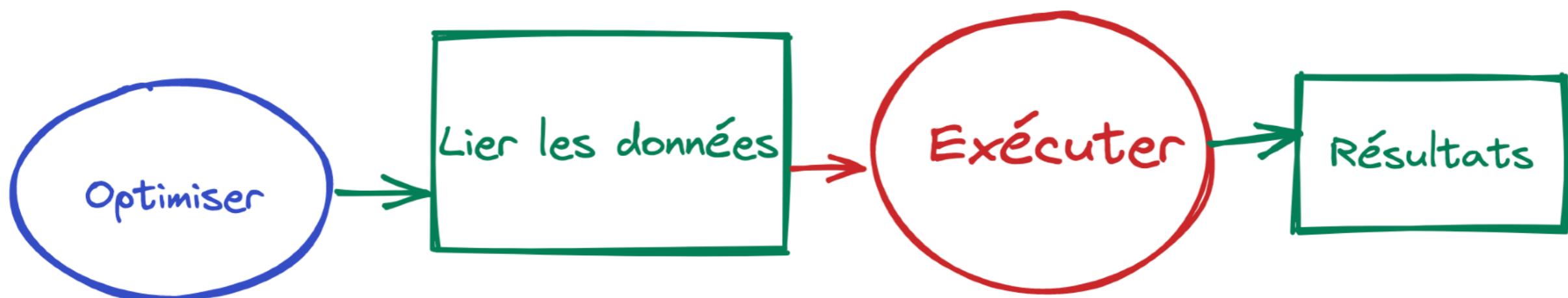
- ▶ On les utilise pour gagner du temps. Car la requête n'est interprétée qu'une seule fois. C'est à dire analysée, compilée et optimisée.
- ▶ L'exécution d'une requête préparée se déroule en deux étapes : la préparation et l'exécution.



# LES REQUÊTES PRÉPARÉES LA SECURITE

---

- ▶ On les utilise pour sécuriser le code. Les paramètres non pas besoin d'être explicitement protégés car le pilote de la base de données le réalise déjà automatiquement.
- ▶ Vous êtes donc protégé des attaques par injection de code SQL !
- ▶ Avant d'exécuter il y a juste à lier les données



# PDO LES METHODES



# LES REQUÊTES PRÉPARÉES LES MÉTHODES

- ▶ **PDO::prepare()** : prépare une requête à l'exécution et retourne un objet **PDOStatement**

## PDO::prepare

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

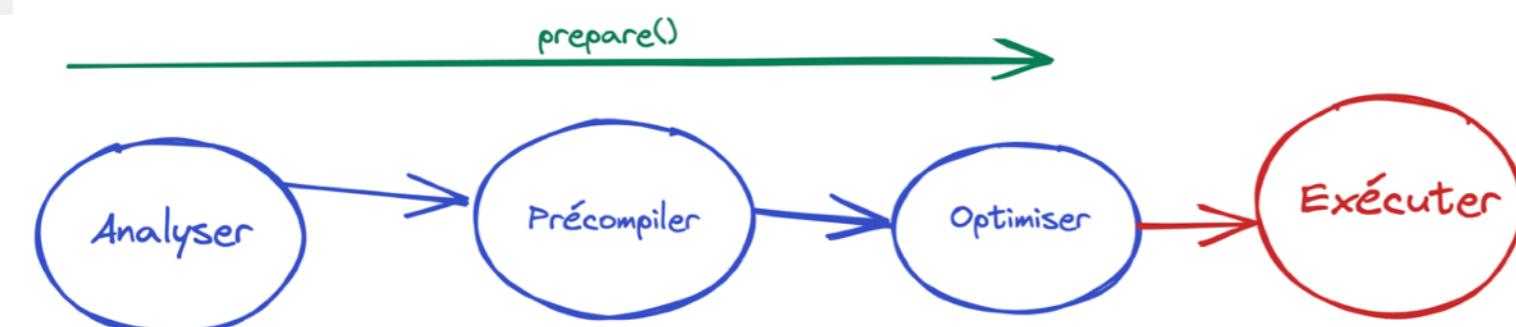
PDO::prepare — Prépare une requête à l'exécution et retourne un objet

### Description

```
public PDO::prepare(string $query, array $options = []): PDOStatement|false
```

Prépare une requête SQL à être exécutée par la méthode [PDOStatement::execute\(\)](#). Le modèle de déclaration peut contenir zéro ou plusieurs paramètres nommés (:nom) ou marqueurs (?) pour lesquels les valeurs réelles seront substituées lorsque la requête sera exécutée. L'utilisation à la fois des paramètres nommés ainsi que les marqueurs est impossible dans un modèle de déclaration ; seul l'un ou l'autre style de paramètre. Utilisez ces paramètres pour lier les entrées utilisateurs, ne les incluez pas directement dans la requête.

Vous devez inclure un marqueur avec un nom unique pour chaque valeur que vous souhaitez passer dans la requête lorsque vous appelez [PDOStatement::execute\(\)](#). Vous ne pouvez pas utiliser un marqueur avec deux noms identiques dans une requête préparée, à moins que le mode émulation ne soit actif.



## PDO

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

# LES REQUÊTES PRÉPARÉES LES MÉTHODES

- ▶ Remarques : On ne peut plus utiliser `query()`
- ▶ On utilisera la méthode **execute()** qui s'applique sur l'objet de type **PDOStatement** retourné par l'exécution de `prepare()`.

## PDOStatement::execute

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

PDOStatement::execute — Exécute une requête préparée

### Description

```
public PDOStatement::execute(?array $params = null): bool
```

Exécute une requête préparée. Si la requête préparée inclut des marqueurs de positionnement, vous pouvez :

- PDOStatement::bindParam() et/ou PDOStatement::bindValue() doit être appelé pour lier des variables ou des valeurs (respectivement) aux marqueurs de paramètres. Les variables liées passent leurs valeurs en entrée et reçoivent les valeurs de sortie, s'il y en a, de leurs marqueurs de positionnement respectifs
- ou passer un tableau de valeurs de paramètres, uniquement en entrée

## PDOStatement

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

# LES REQUÊTES PRÉPARÉES LES MÉTHODES

- ▶ Remarques : On ne peut plus utiliser query()
- ▶ On utilisera la méthode execute() qui s'applique sur l'objet de type **PDOStatement** retourné par l'execution de prepare().
- ▶ Voici un exemple :

```
<?php
// Inclusion du fichier contenant la connexion à la base
include_once('connect.inc.php');

$auteur = 'Ponçon';
$sql = "SELECT * FROM article WHERE auteur =:auteur";
$stmt = $dbh->prepare($sql);
$stmt->execute(array(':auteur'=>$auteur));
echo '<pre>';
while ($row = $stmt->fetch()) {
    echo $row['titre'], '<br>';
    echo $row['auteur'], '<br>';
}
echo '</pre>';
?>
```

# CONSTRUCTION DE LA REQUÊTE

---

- ▶ Paramètres nommés du type  
`$sql2=<< INSERT INTO article (titre, auteur) VALUES (:titre, :auteur) >>;`
- ▶ Avec des points d'interrogations du type  
`$sql3=<< INSERT INTO article (titre, auteur) VALUES (?, ?) >>;`
- ▶ Remarque : on ne peut mixer les deux ! Il faut faire un choix.

# UN EXEMPLE CONCRET



# PRÉPARER LA REQUÊTE

```
<?php
$real ="Darabont";
$req="SELECT titre, realisateur, annee FROM film WHERE realisateur=:monRealisateur";
$result = $bdd->prepare($req);
$result->execute(array(':monRealisateur'=>$real));
if ($result === false)
{
    die("erreur requête !");
}
echo "<h2>".$real."</h2>";
while($film = $result->fetch())
{
    ?>
<tr>
<td><?= $film['titre'] ?></td>
<td><?= date("Y", strtotime($film['annee'])) ?></td>
</tr>
<?php
```

- ▶ On prépare cette requête qui a un paramètre : le nom du réalisateur.  
    \$bdd = new PDO('mysql:host=localhost;dbname=ma\_bdd', 'root', '' );  
    \$req = \$bdd ->prepare("SELECT titre, realisateur, annee FROM film WHERE realisateur=:monRealisateur");  
    \$req->execute(array('monRealisateur'=>\$real));  
    if (\$req === false)  
    {  
        die("erreur requête !");
    }  
    echo "<h2>".\$real."</h2>";  
    while(\$film = \$req->fetch())  
    {  
        ?>  
        <tr>  
        <td><?= \$film['titre'] ?></td>  
        <td><?= date("Y", strtotime(\$film['annee'])) ?></td>  
        </tr>  
    }<?php
- ▶ \$bdd est un objet PDO et la méthode prépare va permettre d'analyser, optimiser cette requête.
- ▶ Si ça se passe bien cette méthode renvoie un objet de type PDOStatement qui est ici récupérer par \$result
- ▶ Si ça se passe mal c'est FALSE qui est retourné

## PLIER LES DONNEES ET EXECUTER

---

- ▶ L'objet **PDOStatement** va permettre d'exécuter la requête via la méthode **execute()** mais il reste à lui passer les données
- ▶ On peut le faire en passant un tableau  
Exemple : `$stmt->execute(array(':titre'=>$titre,  
':auteur'=>$auteur));`
- ▶ Mais on peut faire mieux en passant par **bindParam()** ou **bindValue()**

# LIER LES DONNEES AVEC BINDVALUE

## PDOStatement::bindValue

(PHP 5 >= 5.1.0, PHP 7, PHP 8, PECL pdo >= 1.0.0)

PDOStatement::bindValue — Associe une valeur à un paramètre

### Description

```
public PDOStatement::bindValue(string|int $param, mixed $value, int $type = PDO::PARAM_STR): bool
```

Associe une valeur à un nom correspondant ou à un point d'interrogation (comme paramètre fictif) dans la requête SQL qui a été utilisé pour préparer la requête.

### Liste de paramètres

#### param

Identifiant du paramètre. Pour une requête préparée utilisant les marqueurs, cela sera un nom de paramètre de la forme `:nom`. Pour une requête préparée utilisant les points d'interrogation (comme paramètre fictif), cela sera un tableau indexé numériquement qui commence à la position 1 du paramètre.

#### value

La valeur à associer au paramètre.

#### type

Type de données explicite pour le paramètre utilisant les constantes [PDO::PARAM\\_\\*](#).

### Valeurs de retour

Cette fonction retourne `true` en cas de succès ou `false` si une erreur survient.

# PLIER LES DONNEES ET EXECUTER

---

## Exemples

---

### Exemple #1 Exécute une requête préparée avec des marqueurs nommés

```
<?php
/* Exécute une requête préparée en associant des variables PHP */
$calories = 150;
$couleur = 'rouge';
$sth = $dbh->prepare('SELECT nom, couleur, calories
    FROM fruit
    WHERE calories < :calories AND couleur = :couleur');
$sth->bindValue(':calories', $calories, PDO::PARAM_INT);
$sth->bindValue(':couleur', $couleur, PDO::PARAM_STR);
$sth->execute();
?>
```

# LIER LES DONNEES ET EXECUTER

---

Exemple #2 Exécute une requête préparée avec des points d'interrogation comme paramètre fictif

```
<?php
/* Exécute une requête préparée en associant des variables PHP */
$calories = 150;
$couleur = 'rouge';
$sth = $dbh->prepare('SELECT nom, couleur, calories
    FROM fruit
    WHERE calories < ? AND couleur = ?');
$sth->bindValue(1, $calories, PDO::PARAM_INT);
$sth->bindValue(2, $couleur, PDO::PARAM_STR);
$sth->execute();
?>
```

# LIER LES DONNEES AVEC BINDPARAM

## PDOStatement::bindParam

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

PDOStatement::bindParam — Lie un paramètre à un nom de variable spécifique

### Description

```
public PDOStatement::bindParam(  
    string|int $param,  
    mixed &$var,  
    int $type = PDO::PARAM_STR,  
    int $maxLength = 0,  
    mixed $driverOptions = null  
) : bool
```

Lie une variable PHP à un marqueur nommé ou interrogatif correspondant dans une requête SQL utilisée pour préparer la requête. Contrairement à [PDOStatement::bindValue\(\)](#), la variable est liée en tant que référence et ne sera évaluée qu'au moment de l'appel à la fonction [PDOStatement::execute\(\)](#).

La plupart des paramètres sont des paramètres d'entrées, et sont utilisés en lecture seule pour construire la requête (mais peuvent néanmoins être transtypé en fonction de [data\\_type](#)). Quelques drivers supportent l'invocation de procédures stockées qui retournent des données en tant que paramètres de sortie, et quelques autres en tant que paramètres entrées / sorties qui sont envoyés ensemble et sont mis à jour pour les recevoir.

### Liste de paramètres

#### param

Identifiant. Pour une requête préparée utilisant des marqueurs nommés, ce sera le nom du paramètre sous la forme `:name`. Pour une requête préparée utilisant les marqueurs interrogatifs, ce sera la position indexé +1 du paramètre.

#### var

Nom de la variable PHP à lier au paramètre de la requête SQL.

# LIER LES DONNEES AVEC BINDPARAM

Exemple #1 Exécution d'une requête préparée avec des emplacements nommés

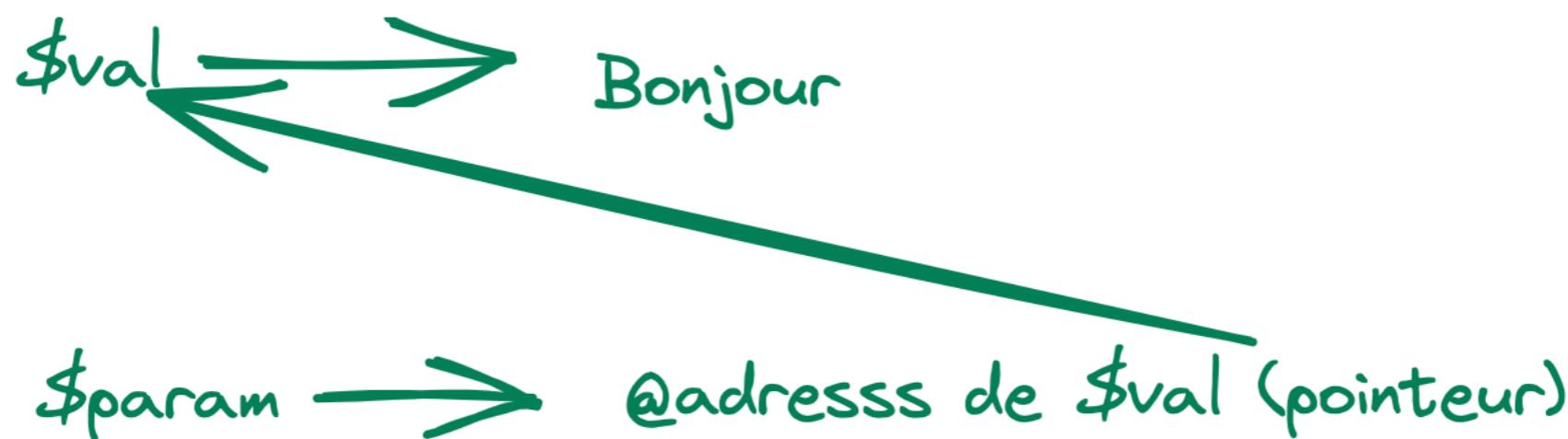
```
<?php
/* Exécution d'une requête préparée en liant des variables PHP */
$calories = 150;
$couleur = 'rouge';
$sth = $dbh->prepare('SELECT nom, couleur, calories
    FROM fruit
    WHERE calories < :calories AND couleur = :couleur');
$sth->bindParam(':calories', $calories, PDO::PARAM_INT);
$sth->bindParam(':couleur', $couleur, PDO::PARAM_STR, 12);
$sth->execute();
?>
```

Exemple #2 Exécution d'une requête préparée avec des marqueurs de positionnement

```
<?php
/* Exécution d'une requête préparée en liant des variables PHP */
$calories = 150;
$couleur = 'rouge';
$sth = $dbh->prepare('SELECT nom, couleur, calories
    FROM fruit
    WHERE calories < ? AND couleur = ?');
$sth->bindParam(1, $calories, PDO::PARAM_INT);
$sth->bindParam(2, $couleur, PDO::PARAM_STR, 12);
$sth->execute();
?>
```

## PLIER LES DONNEES ET EXECUTER

- ▶ La différence entre **bindValue()** et **bindParam()** paraît minime et pourtant c'est très différent !
- ▶ Dans le cas de **bindValue()** on associe une valeur à un paramètre. Juste à ce moment là.
- ▶ Alors que **bindParam()** lie une variable à un paramètre. C'est une référence vers un paramètre.



# PLIER LES DONNEES ET EXECUTER

```
<?php
// Inclusion du fichier contenant la connexion à la base
include_once('connect.inc.php');

$sql = 'INSERT INTO article (titre, auteur)
        VALUES (:titre , :auteur)';

$stmt = $dbh->prepare($sql);

$titre = 'Memento PHP MySQL';
$auteur = 'Ponçon' ;

$stmt->bindParam(':auteur',$auteur);
$stmt->bindParam(':titre',$titre);
$stmt->execute();
// Un premier enregistrement a été inséré

$titre = 'Best practices PHP 5' ;
$stmt->execute();
?>
```

Exemple ici si on avait utiliser **bindValue()** en lieu et place du **bindParam()** le deuxième **execute()** aurait donné le même résultat que le premier.

Ce qui ne sera pas le cas ici !

## LIBÉRER LES REQUÊTES PRÉPARÉES

- ▶ On ne peut avoir deux requêtes préparées en même temps.
- ▶ Si vous voulez en préparer une autre il faut libérer la première.
- ▶ Il peut être intéressant également de libérer une ressource dont on n'a plus besoin.
- ▶ C'est toujours pareil en PHP on donne la valeur NULL au pointeur vers l'objet ( je vous rappelle que les variable et la variable \$stmt ne contient pas l'objet mais une référence à celui-ci )  
Exemple : `$stmt = NULL;`
- ▶ L'objet n'étant plus référencé par une variable PHP peut libérer cet espace mémoire !



MERCI POUR  
VOTRE ATTENTION

