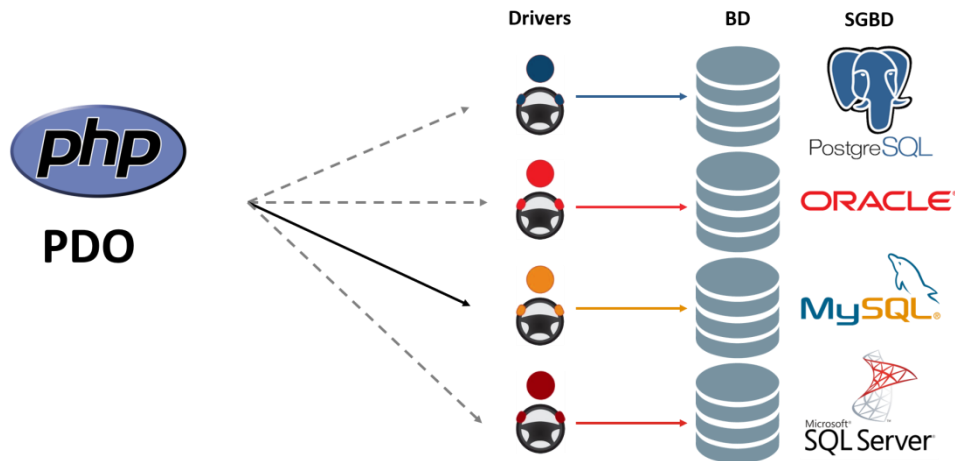


## 1 PDO : Php Data Object

**PDO** est une **extension PHP** qui permet d'utiliser une base de données en programmant avec un **style orienté objet**, et surtout qui permet de **s'affranchir du SGBD**.

**PDO** n'utilise pas des fonctions au nom trop explicite comme **mysql\_query()** ou **sqlite\_query()**, ce qui facilite grandement la migration d'un SGBD à l'autre, voire l'utilisation simultanée ou alternée de **plusieurs SGBD** avec **le même code PHP**.

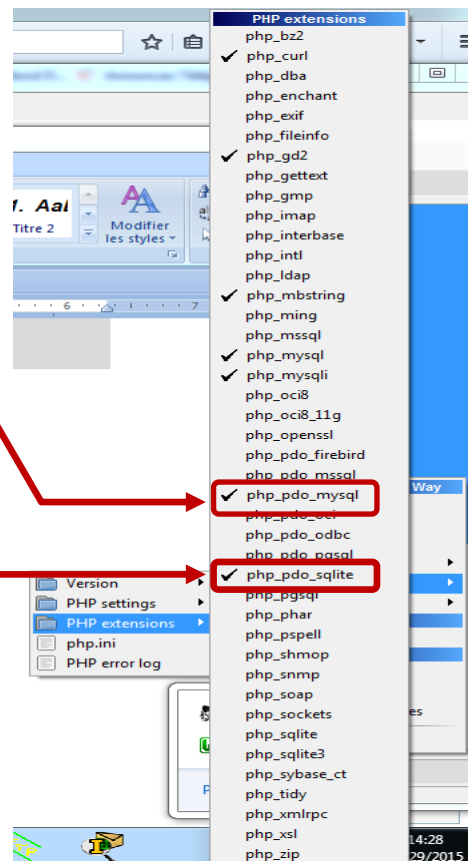


**PDO est une solution d'abstraction de BDD en PHP.**

**Chaque pilote** est associé à une extension qu'il faut penser à activer dans le fichier **php.ini** afin de pouvoir l'utiliser.

- Firebird/Interbase 6 ;
- FreeTDS / Microsoft SQL Server / Sybase (à ne pas utiliser, module non tenu à jour par Microsoft) ;
- IBM DB2 ;
- IBM Informix Dynamic Server ;
- **MySQL 3.x, 4.x, 5.x (optimisé pour 4.1 et supérieur) ;**
- ODBC v3 (IBM DB2 unixODBC et win32 ODBC) ;
- Oracle Call Interface ;
- PostgreSQL ;
- **SQLite 3 et SQLite 2.**

### Activation de PDO pour MySql



## 2 La connexion

La première chose à faire pour utiliser **PDO** est bien sûr **d'établir une connexion à une base de données** :

```
<?php
try
{
    // Connexion à la base de données
    $pdo = new PDO('mysql:host=localhost;dbname=test', 'user', 'passwd');

    // Configuration facultative de la connexion
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION); // les erreurs
    lanceront des exceptions

    // utiliser la connexion ici
    ... Les requêtes SQL vont s'écrire ici .....
    // et maintenant, fermer la connexion !
    $pdo = null;
}

catch(PDOException $e)
{
    $msg = 'ERREUR PDO dans ' . $e->getFile() . ' ligne. ' . $e->getLine() . ' : ' .
    $e->getMessage();
    die($msg);
}

?>
```

## 3 Les requêtes non préparées

PDO fait la distinction entre deux formes de requêtes : "**exec**" et "**query**".

### 3.1 La commande **exec** (ECRITURE)

Elle est valable pour les **requêtes SQL d'écriture**, soit : INSERT, UPDATE et DELETE

#### 3.1.1 INSERT

On ajoute un enregistrement au sein de la BDD :

```
$pdo->exec("INSERT INTO membres (champ_login, champ_mdp)
VALUES ('login', 'mot_de_passe')");
```

#### 3.1.2 UPDATE

On met à jour un enregistrement au sein de la BDD :

```
$pdo->exec("UPDATE membres SET champ_login='login', champ_mdp='mot_de_passe'
WHERE champ_id_membre = 'id_membre'");
```

#### 3.1.3 DELETE

On supprime un enregistrement au sein de la BDD :

```
$pdo->exec("DELETE FROM membres WHERE champ_id_membre='id_membre'");
```

### 3.2 La commande **query** (LECTURE)

Elle est valable pour les **requêtes SQL de lecture**, soit : SELECT

On recherche une entrée dans la BDD :

```
$resultats=$pdo->query("SELECT membre FROM table WHERE
champ_id_membre='id_membre'");
```

## 4 Le FETCH (parcourir les réponses d'une requête query)

Récupère un enregistrement depuis un jeu de résultats associé à l'objet *PDOStatement*.

Considérons cet exemple de table "user" au sein d'une base de données "pdodatabase"

Id	first_name	last_name	email
1	jean	bon	jean.bon@free.fr
2	jessica	trise	jessica.trise@outlook.fr
3	lara	clette	lara.clette@orange.fr

Il existe essentiellement trois options:

### 4.1 PDO::FETCH\_ASSOC [ ]:

retourne un tableau indexé par le nom de la colonne comme retourné dans le jeu de résultats

```
<?php
// fetch as associative array
$dns = 'mysql:host=localhost;dbname=pdodatabase';
$user = 'root';
$pass = '';
// la connexion à la base de données
try {
    $pdo = new PDO($dns, $user, $pass, array( PDO::ATTR_PERSISTENT => true));
}
catch (Exception $e) {
    die("Impossible de se connecter: " . $e->getMessage());
}
try {
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    // la requête SQL
    $stmt = $pdo->query("SELECT id,first_name,last_name,email FROM user");
    // affichage du résultat
    echo '<table border="1">';
    echo '<tr><th>Id</th><th>First Name</th><th>Last Name</th><th>Email</th></tr>';

    // access record fields with name
    while ($r = $stmt->fetch(PDO::FETCH_ASSOC))
    {
        echo '<tr>';
        echo '<td>' . $r['id'] . '</td>';
        echo '<td>' . $r['first_name'] . '</td>';
        echo '<td>' . $r['last_name'] . '</td>';
        echo '<td>' . $r['email'] . '</td>';
        echo '</tr>';
    }

    echo '</table>';
}
catch (Exception $e) { echo "Failed: " . $e->getMessage(); }
?>
```

Id	First Name	Last Name	Email
1	jean	bon	jean.bon@free.fr
2	jessica	trise	jessica.trise@outlook.fr
3	lara	clette	lara.clette@orange.fr

## 4.2 PDO::FETCH\_NUM [ ]:

- retourne un tableau indexé par le numéro de la colonne comme elle est retournée dans votre jeu de résultat, commençant à 0

```
<?php
// fetch as NUM
$dns = 'mysql:host=localhost;dbname=pdoDataBase';
$user = 'root';
$pass = '';

try {
    $pdo = new PDO($dns, $user, $pass, array( PDO::ATTR_PERSISTENT => true));
}
catch (Exception $e) {
    die("Impossible de se connecter: " . $e->getMessage());
}
try {
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    // les requêtes SQL
    $stmt = $pdo->query("SELECT id,first_name,last_name,email FROM user");

    // affichage du résultat
    echo '<table border="1">';
    echo '<tr><th>Id</th><th>First Name</th><th>Last Name</th><th>Email</th></tr>';

    // access record fields with index
    while ($r = $stmt->fetch(PDO::FETCH_NUM))
    {
        echo '<tr>';
        echo '<td>' . $r[0] . '</td>';
        echo '<td>' . $r[1] . '</td>';
        echo '<td>' . $r[2] . '</td>';
        echo '<td>' . $r[3] . '</td>';
        echo '</tr>';
    }

    echo '</table>';
}
catch (Exception $e) {
    echo "Failed: " . $e->getMessage();
}
?>
```

Id	First Name	Last Name	Email
1	jean	bon	jean.bon@free.fr
2	jessica	trise	jessica.trise@outlook.fr
3	lara	clette	lara.clette@orange.fr

## 4.3 PDO::FETCH\_OBJ:

- retourne un **objet anonyme** avec les noms de propriétés qui correspondent aux noms des colonnes retournés dans le jeu de résultats

```
<?php
// fetch as Object
$dns = 'mysql:host=localhost;dbname=pdoDataBase';
$user = 'root';
$pass = '';

try {
    $pdo = new PDO($dns, $user, $pass, array( PDO::ATTR_PERSISTENT => true));
}
catch (Exception $e) {
    die("Impossible de se connecter: " . $e->getMessage());
}

try {
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $pdo->beginTransaction(); //initie la transaction, désactive autocommit

    // les requêtes SQL
    $stmt = $pdo->query("SELECT id,first_name,last_name,email FROM user");

    // affichage du résultat
    echo '<table border="1">';
    echo '<tr><th>Id</th><th>First Name</th><th>Last Name</th><th>Email</th></tr>';

    // access record fields as object
    while ($r = $stmt->fetch(PDO::FETCH_OBJ))
    {
        echo '<tr>';
        echo '<td>' . $r->id . '</td>';
        echo '<td>' . $r->first_name . '</td>';
        echo '<td>' . $r->last_name . '</td>';
        echo '<td>' . $r->email . '</td>';
        echo '</tr>';
    }

    echo '</table>';
}
catch (Exception $e) {
    echo "Failed: " . $e->getMessage();
}

?>
```

Id	First Name	Last Name	Email
1	jean	bon	jean.bon@free.fr
2	jessica	trise	jessica.trise@outlook.fr
3	lara	clette	lara.clette@orange.fr

## 4.4 FetchAll :

Il s'agit ici d'une variante du fetch précédent.

- **fetch** récupère **une ligne à la fois**,
- alors que **fetchAll** récupères **autant de lignes qu'il y a de réponses**

On peut alors parcourir cet ensemble de résultats à l'aide d'un foreach

```
<?php
// fetch as Object
$dns  = 'mysql:host=localhost;dbname=pdoDataBase';
$user = 'root';
$pass = '';

try {
    $pdo = new PDO($dns, $user, $pass, array( PDO::ATTR_PERSISTENT => true));
}
catch (Exception $e) {
    die("Impossible de se connecter: " . $e->getMessage());
}
try
{
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    // les requêtes SQL
    $stmt = $pdo->query("SELECT id,first_name,last_name,email FROM user");

    // affichage du résultat
    echo '<table border="1">';
    echo '<tr><th>Id</th><th>First Name</th><th>Last Name</th><th>Email</th></tr>';

    // access record fields as object
    $results = $stmt->fetchAll(PDO::FETCH_OBJ) ;

    foreach($results as $r)
    {
        echo '<tr>';
        echo '<td>' . $r->id . '</td>';
        echo '<td>' . $r->first_name . '</td>';
        echo '<td>' . $r->last_name . '</td>';
        echo '<td>' . $r->email . '</td>';
        echo '</tr>';
    }

    echo '</table>';
    $pdo->commit(); // valide la transaction (on exécute les requêtes)
}
catch (Exception $e) {
    echo "Failed: " . $e->getMessage();
}
?>
```

Id	First Name	Last Name	Email
1	jean	bon	jean.bon@free.fr
2	jessica	trise	jessica.trise@outlook.fr
3	lara	clette	lara.clette@orange.fr

## 5 Les requêtes préparées

L'utilisation de **requêtes préparées** offrira les avantages suivants :

- + impose une certaine rigueur de programmation
- + optimise le temps d'exécution requis pour les requêtes exécutées plus d'une fois
- + offre une plus grande sécurité au niveau des requêtes (risque d'injection de code)



Deux fonctions permettent de préparer les requêtes :

### 5.1 bindParam (avec ?)

- + PDOStatement::**bindParam()** va remplacer telle **étiquette** par telle **variable**,

Un exemple :

```
<?php
// fetch as Object
$dns = 'mysql:host=localhost;dbname=pdoDataBase';
$user = 'root';
$pass = '';

try {
    $pdo = new PDO($dns, $user, $pass, array( PDO::ATTR_PERSISTENT => true));
}
catch (Exception $e) {
    die("Impossible de se connecter: " . $e->getMessage());
}

try {
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // une requête préparée
    $stmt = $pdo->prepare("INSERT INTO user (first_name,last_name,email)
                           VALUES (?, ?, ?)");

    $stmt->bindParam(1, $firstname);
    $stmt->bindParam(2, $lastname);
    $stmt->bindParam(3, $email);

    // insertion d'une ligne
    $firstname = "jean";
    $lastname = "meurdesoif";
    $email = "jean.meurdesoif@hotmail.fr";
    $stmt->execute(); // a noter qu'exec s'est transformé ici en execute

    // insertion d'une autre ligne avec d'autres valeurs
    $firstname = "ray";
    $lastname = "defesse";
    $email = "ray.defesse@voila.fr";
    $stmt->execute(); // a noter qu'exec s'est transformé ici en execute
```

```
// les requêtes SQL
$stmt = $pdo->query("SELECT id,first_name,last_name,email FROM user");

// affichage du résultat
echo '<table border="1">';
echo '<tr><th>Id</th><th>First Name</th><th>Last Name</th><th>Email</th></tr>';

// access record fields as object
$results = $stmt->fetchAll(PDO::FETCH_OBJ);

foreach($results as $r)
{
    echo '<tr>';
    echo '<td>' . $r->id . '</td>';
    echo '<td>' . $r->first_name . '</td>';
    echo '<td>' . $r->last_name . '</td>';
    echo '<td>' . $r->email . '</td>';
    echo '</tr>';
}
echo '</table>';
}
catch (Exception $e)
{
    echo "Failed: " . $e->getMessage();
}
?>
```

Id	First Name	Last Name	Email
1	jean	bon	jean.bon@free.fr
2	jessica	trise	jessica.trise@outlook.fr
3	lara	clette	lara.clette@orange.fr
7	jean	meurdesoif	jean.meurdesoif@hotmail.fr
8	ray	defesse	ray.defesse@voila.fr

## 5.2 BindValue (avec ?)

✚ PDOStatement::bindValue() va remplacer telle étiquette par telle valeur

Un exemple :

```
<?php
// fetch as Object
$dns = 'mysql:host=localhost;dbname=pdoDataBase';
$user = 'root';
$pass = '';

try {
    $pdo = new PDO($dns, $user, $pass, array( PDO::ATTR_PERSISTENT => true));
}
catch (Exception $e) {
    die("Impossible de se connecter: " . $e->getMessage());
}
try {
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    // une requête préparée
    $stmt = $pdo->prepare("INSERT INTO user (first_name,last_name,email)
                        VALUES (?, ?, ?)");

    $stmt->bindValue(1, "jean", PDO::PARAM_STR);
    $stmt->bindValue(2, "meurdesoif", PDO::PARAM_STR);
```



# PDO (MySQL)

```
$stmt->bindValue(3, "jean.meurdesoif@hotmail.fr", PDO::PARAM_STR);
$stmt->execute(); // a noter que exec s'est transformé ici en execute

// insertion d'une autre ligne avec d'autres valeurs
$stmt->bindValue(1, "ray", PDO::PARAM_STR);
$stmt->bindValue(2, "defesse", PDO::PARAM_STR);
$stmt->bindValue(3, "ray.defesse@voila.fr", PDO::PARAM_STR);
$stmt->execute(); // a noter que exec s'est transformé ici en execute

// les requêtes SQL
$stmt = $pdo->query("SELECT id,first_name,last_name,email FROM user");

// affichage du résultat
echo '<table border="1">';
echo '<tr><th>Id</th><th>First Name</th><th>Last Name</th><th>Email</th></tr>';

// access record fields as object
$results = $stmt->fetchAll(PDO::FETCH_OBJ);
foreach($results as $r)
{
    echo '<tr>';
    echo '<td>' . $r->id . '</td>';
    echo '<td>' . $r->first_name . '</td>';
    echo '<td>' . $r->last_name . '</td>';
    echo '<td>' . $r->email . '</td>';
    echo '</tr>';
}
echo '</table>';
}
catch (Exception $e) {
    echo "Failed: " . $e->getMessage();
}
?>
```

Id	First Name	Last Name	Email
1	jean	bon	jean.bon@free.fr
2	jessica	trise	jessica.trise@outlook.fr
3	lara	clette	lara.clette@orange.fr
7	jean	meurdesoif	jean.meurdesoif@hotmail.fr
8	ray	defesse	ray.defesse@voila.fr

## 5.3 BindParam vs BindValue (pour information)

Voici un élément de comparaison entre **bindParam** et **bindValue** :

```
$stmt = $db->prepare('SELECT * FROM tableName WHERE leChamp = :laValeur');
$var = 'foo';

$stmt->bindValue(':laValeur', $var);
$var = 'bar';
$stmt->execute();
```


The above executes like **SELECT \* FROM tableName WHERE leChamp = 'foo'**;

```
$stmt = $db->prepare('SELECT * FROM tableName WHERE leChamp = :laValeur');
$var = 'foo';
$stmt->bindParam(':laValeur', $var);
$var = 'bar';
$stmt->execute();
```

The above executes like **SELECT \* FROM tableName WHERE leChamp = 'bar'**

## 5.4 Alternative au bind ( avec :var)

L'opération **bindParam** (vs **bindValue**) peut prendre également cette autre forme :



```
<?php
$nom='toto';
$age=89;
$boolexemple=true;

//Première alternative
$rep=$bdd->prepare('INSERT INTO maTable VALUES("", :nom, :age, :boolexemple)');

$rep->bindParam('nom', $nom, PDO::PARAM_STR);
$rep->bindParam('age', $age, PDO::PARAM_INT);
$rep->bindParam('boolexemple', $boolexemple, PDO::PARAM_BOOL);
$rep->execute();

//Seconde alternative
$rep=$bdd->prepare('INSERT INTO maTable VALUES("", :nom, :age, :boolexemple)');

$rep->execute(array(
    'nom'=>$nom,
    'age'=>$age,
    'boolexemple'=>$boolexemple,
));
?>
```

## 5.5 Quelques exemples

### 5.5.1 exemple

```
<?php
/* Exécute 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);
$sth->execute();
?>
```

### 5.5.2 exemple

```
$pdo = new PDO('mysql:host=127.0.0.1;dbname=nomdelabase', 'user', 'motdepasse');
$stmt = $pdo->prepare('SELECT label, description FROM books WHERE id = :id',
    PDO::PARAM_STR);
$stmt->bindValue(':id', 3, PDO::PARAM_INT);
$stmt->execute();
var_dump($stmt->fetch());
```

### 5.5.3 exemple

```
$stmt = $pdo->query("SELECT * FROM users LIMIT ?, ?");
$stmt->execute([$limit, $offset]);

while ($row = $stmt->fetch()) {
    echo $row['name']."<br />\n";
}
```

## 5.5.4 exemple

```
$stmt = $pdo->prepare("SELECT * FROM users LIMIT :limit, :offset");
$stmt->execute(['limit' => $limit, 'offset' => $offset]);

$data = $stmt->fetchAll();
// and somewhere later:
foreach ($data as $row) {
    echo $row['name']."<br />\n";
}
```

## 5.5.5 exemple

```
$pdo = new PDO('mysql:host=127.0.0.1;dbname=nomdelabase', 'user', 'motdepasse');
$stmt = $pdo->prepare('INSERT INTO books (label, description) VALUES (:label, :description)');
$stmt->bindValue(':label', 'Harry Potter', PDO::PARAM_STR);
$stmt->bindValue(':description', 'Tome numéro 7 - Partie 2', PDO::PARAM_STR);
$stmt->execute();
```

## 5.5.6 exemple

```
$pdo = new PDO('mysql:host=127.0.0.1;dbname=nomdelabase', 'user', 'motdepasse');
$stmt = $pdo->prepare('UPDATE books SET description = :description WHERE id = :id');
$stmt->bindValue(':description', 'Tome numéro 8 - Voldemort le retour', PDO::PARAM_STR);
$stmt->bindValue(':id', 1, PDO::PARAM_INT);
$stmt->execute();
```

## 5.5.7 exemple

```
<?php
/* Exécute une requête préparée en passant un tableau de valeurs */
$calories = 150;
$couleur = 'rouge';
$stmt = $dbh->prepare('SELECT nom, couleur, calories FROM fruit WHERE calories < :calories AND couleur = :couleur');
$stmt->execute(array(':calories' => $calories, ':couleur' => $couleur));
?>
```

## 5.5.8 exemple

```
<?php
/* Exécute une requête préparée en passant un tableau de valeurs */
$calories = 150;
$couleur = 'rouge';
$stmt = $dbh->prepare('SELECT nom, couleur, calories FROM fruit WHERE calories < ? AND couleur = ?');
$stmt->execute(array($calories, $couleur));
?>
```

## 5.5.9 exemple

```
$nom = "Flo";
$prenom = "Dechand";
$adresse = "Rue des Moulins";
$ville = "Marseille";
$cp = 13001;
$pays = "France";
$mail = "flodc@gmail.com";
```

```
$sth = $dbco->prepare("INSERT INTO
Clients(Nom,Prenom,Adresse,Ville,Codepostal,Pays,Mail) VALUES (:nom, :prenom,
:adresse, :ville, :cp, :pays, :mail)
");
$sth->execute(array(':nom' => $nom,':prenom' => $prenom,':adresse' => $adresse,
':ville' => $ville,':cp' => $cp,':pays' => $pays,':mail' => $mail));
```

## 6 Une classe PHP de connexion à la BDD

Toutes ces notions peuvent bien entendu être **encapsulées au sein d'une classe en PHP**.

### 6.1 La classe (fichier crudPDO.php)

Cette classe va permettre de se **connecter à une base de données et d'y faire des requêtes**

```
<?php
class CrudPDO {
    private $pdo;
    public function __construct($host, $dbname, $username, $password) {
        try {
            // Create a PDO instance for database connection
            $this->pdo = new PDO("mysql:host=$host;dbname=$dbname;charset=utf8", $username,
$password);
            // Set PDO to throw exceptions on errors
            $this->pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        } catch (PDOException $e) {
            throw new Exception("Database connection failed: " . $e->getMessage());
        }
    }

    public function create($table, $data) {
        try {
            // Build the SQL INSERT query
            $fields = implode(', ', array_keys($data));
            $values = ':' . implode(':', array_keys($data));
            $sql = "INSERT INTO $table ($fields) VALUES ($values)";
            // Prepare the statement
            $stmt = $this->pdo->prepare($sql);
            // Bind parameters and execute the statement
            $stmt->execute($data);
            return true;
        } catch (PDOException $e) {
            return false;
        }
    }

    public function read($table) {
        try {
            // Build the SQL SELECT query
            $sql = "SELECT * FROM $table";
            // Execute the query and fetch all records
            $stmt = $this->pdo->query($sql);
            $data = $stmt->fetchAll(PDO::FETCH_ASSOC);
            return $data;
        } catch (PDOException $e) {
            return [];
        }
    }
}
```

```
public function readCustom($sql, $params = []) {
    try {
        // Prepare the custom SQL query
        $stmt = $this->pdo->prepare($sql);

        // Bind parameters and execute the statement
        $stmt->execute($params);

        // Fetch all records
        $data = $stmt->fetchAll(PDO::FETCH_ASSOC);
        return $data;
    } catch (PDOException $e) {
        return [];
    }
}

public function update($table, $id, $data) {
    try {
        // Build the SQL UPDATE query
        $set = implode(', ', array_map(function ($key) {
            return "$key = :$key";
        }, array_keys($data)));
        $sql = "UPDATE $table SET $set WHERE id = :id";

        // Add the id to the data array
        $data['id'] = $id;

        // Prepare the statement
        $stmt = $this->pdo->prepare($sql);

        // Bind parameters and execute the statement
        $stmt->execute($data);
        return true;
    } catch (PDOException $e) {
        return false;
    }
}

public function delete($table, $id) {
    try {
        // Build the SQL DELETE query
        $sql = "DELETE FROM $table WHERE id = :id";

        // Prepare the statement
        $stmt = $this->pdo->prepare($sql);

        // Bind parameters and execute the statement
        $stmt->execute(['id' => $id]);
        return true;
    } catch (PDOException $e) {
        return false;
    }
}
?>
```

## 6.2 Quelques explications sur l'instruction "implode"

Voici une illustration de l'instruction "**implode**" en php :

### 6.2.1 Soit les données suivantes

```
$data = [  
    'name'    => 'John',  
    'age'     => 30,  
    'city'    => 'New York',  
];
```

### 6.2.2 appel à la méthode "create"

lorsque l'on fait appel à la méthode "**create**" en lui passant en paramètres **le tableau associatif précédent** (\$data) :

```
public function create(Stable, Sdata) {  
    try {  
        // Build the SQL INSERT query  
        Sfields  = implode(',', array_keys(Sdata));  
        Svalues  = ':' . implode(':', array_keys(Sdata));  
        Ssql = "INSERT INTO Stable (Sfields) VALUES (Svalues)";  
    }  
}
```

"name, age, city"

":name, :age, :city"

INSERT INTO users (name, age, city) VALUES (:name, :age, :city)

## 6.3 Un premier exemple basique d'utilisation au sein d'un script php

```
<?php  
  
require_once 'CrudPDO.php';  
  
// Database connection details  
$host      = 'localhost';  
$dbname    = 'your_database_name';  
$username  = 'your_username';  
$password  = 'your_password';  
$crud      = new CrudPDO($host, $dbname, $username, $password);  
  
// Create a new record  
$newUser = [  
    'username' => 'john_doe',  
    'email'    => 'john.doe@example.com',  
];  
  
if ($crud->create('users', $newUser)) {  
    echo "New user created successfully!";  
} else {  
    echo "Error creating user.";  
}
```

```
// Read records inside a single table
$users = $crud->read('users');
foreach ($users as $user) {
    echo "ID: " . $user['id'] . "<br>";
    echo "Username: " . $user['username'] . "<br>";
    echo "Email: " . $user['email'] . "<br>";
    echo "<hr>";
}

// Update a record
$userIdToUpdate = 1 ; // Replace with the ID of the user you want to update
$updatedUserData = [
    'username' => 'updated_username',
    'email' => 'updated_email@example.com',
];
if ($crud->update('users', $userIdToUpdate, $updatedUserData)) {
    echo "User updated successfully!";
} else {
    echo "Error updating user.";
}

// Delete a record
$userIdToDelete = 2 ; // Replace with the ID of the user you want to delete
if ($crud->delete('users', $userIdToDelete)) {
    echo "User deleted successfully!";
} else {
    echo "Error deleting user.";
}
?>
```

## 6.4 Un second exemple plus sophistiqué (login avec try...catch)

```
<?php

require_once 'CrudPDO.php'; // Include the CrudPDO class file

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    // Get user input from the login form
    $username = $_POST['username'];
    $password = $_POST['password'];

    // Initialize the CrudPDO class with your database credentials
    $db = new CrudPDO('your_host', 'your_dbname', 'your_username', 'your_password');

    try {
        // Write a custom SQL query to retrieve user data based on the provided username
        $sql = "SELECT * FROM users WHERE username = :username";
        $params = array(':username' => $username);

        // Use the readCustom method to execute the query
        $userData = $db->readCustom($sql, $params);

        if (!empty($userData)) {
            // User with the provided username found
            $storedPassword = $userData[0]['password']; // Assuming you have a 'password' column in your
            users table
        }
    }
}
```

```
// Verify the password
if (password_verify($password, $storedPassword)) {
    // Password is correct, user is authenticated
    echo "Login successful!";
    // You can set session variables or perform other actions here.
} else {
    echo "Incorrect password";
}
} else {
    // User with the provided username not found
    echo "User not found";
}
}
catch (Exception $e) {
    // Handle database connection or query execution errors
    echo "Database error: " . $e->getMessage();
}
?>
```