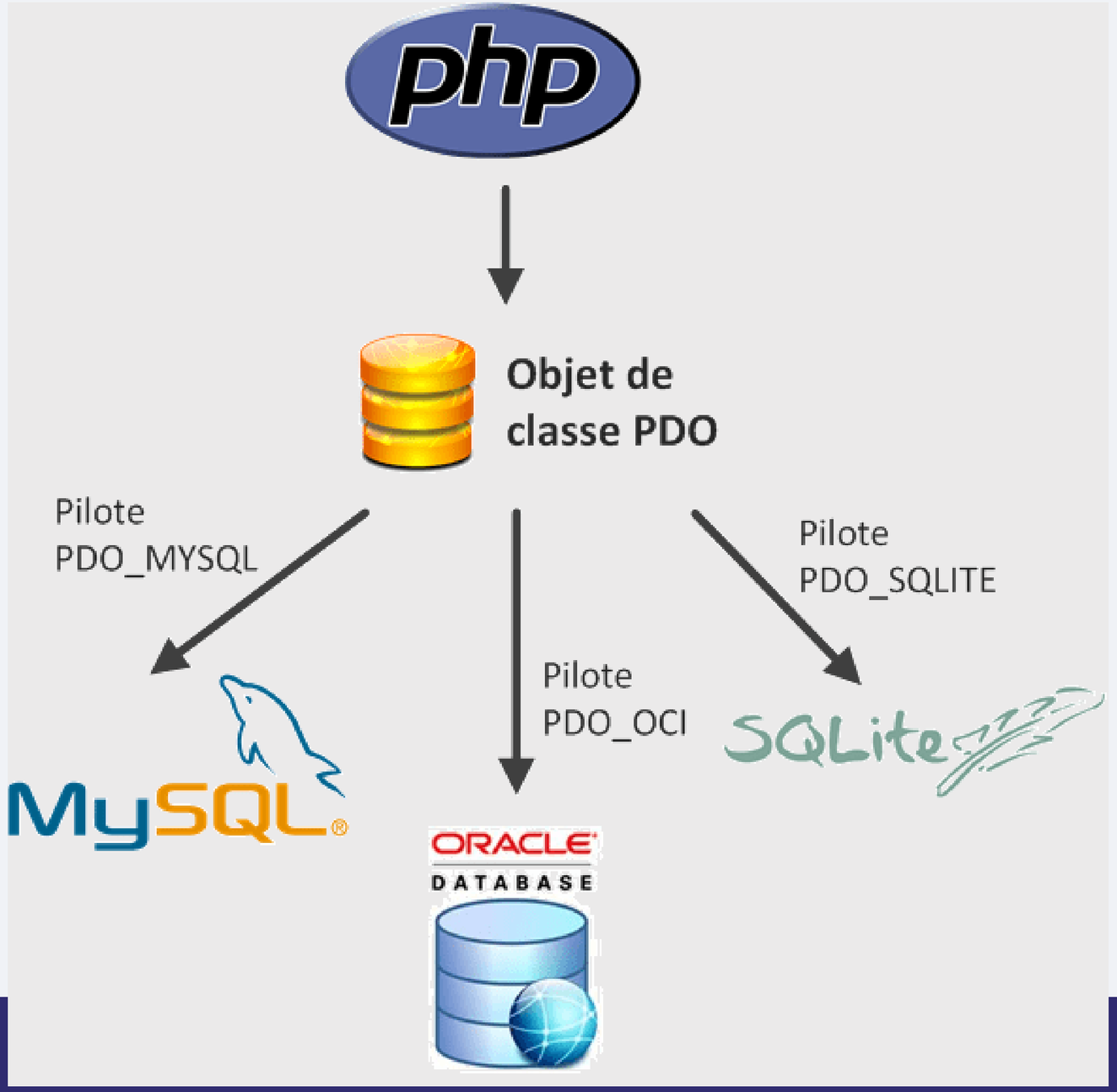
Tutorial de PDO

01

QUE ES PDO

PDO significa PHP Data Objects, Objetos de Datos de PHP, una extensión para acceder a bases de datos.

PDO permite acceder a diferentes sistemas de bases de datos con un controlador específico (MySQL, SQLite, Oracle...) mediante el cual se conecta.


php
Objet de
classe PDO
Pilote
PDO_MYSQL
Pilote
PDO_OCI
Pilote
PDO_SQLITE
MySQL
ORACLE
DATABASE
SQLite

EJEMPLO DE ORACLE

```php
<?php
$param = $_POST;
$db_username = "youusername";
$db_password = "yourpassword";
$db = "oci:dbname=yoursid";
$conn = new PDO($db,$db_username,$db_password);
$name = $param['module'];
$file = $param['file'];
$stmt = $conn->exec("INSERT INTO AL_MODULE (AL_MODULENAME, AL_MODULEFILE) VALUES ('$name', '$file')");
```

solución

Enumera de 1 a 3 soluciones
que tu empresa propone.

EJEMPLO DE SQL SERVER

```php
$contraseña = "hunter2";
$usuario = "usuario";
$nombreBaseDeDatos = "pruebas_parzibyte";
# Puede ser 127.0.0.1 o el nombre de tu equipo; o la IP de un servidor remoto
$rutaServidor = "127.0.0.1";
try {
    $base_de_datos = new PDO("sqlsrv:server=$rutaServidor;database=$nombreBaseDeDatos", $usuario, $contraseña);
    $base_de_datos->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (Exception $e) {
    echo "Ocurrió un error con la base de datos: " . $e->getMessage();
}
```

EJEMPLO DE MYSQL

```php
<?php
class Database
{
    public static function StartUp()
    {
        $pdo = new PDO('mysql:host=localhost;dbname=pruebas;charset=utf8', 'root', '');
        $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        return $pdo;
    }
}
```

Similitudes a Mysqli

```php
public static function conectar()
{
    $pdo = new PDO('mysql:host=' . DB_HOST . ';dbname=' . DB . ';charset=' . CHARSET, DB_USER, DB_PASS);
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    return $pdo;
}

public static function conectarMysqli()
{
    $mysqli = new mysqli(DB_HOST, DB_USER, DB_PASS, DB);
    $mysqli->set_charset(CHARSET);
    return $mysqli;
}
```

Errores en PDO

```
$dbh->setAttribute(PDO::ATTRR_ERRMODE, PDO::ERRMODE_SILENT);
$dbh->setAttribute(PDO::ATTRR_ERRMODE, PDO::ERRMODE_WARNING);
$dbh->setAttribute(PDO::ATTRR_ERRMODE, PDO::ERRMODE_EXCEPTION);
```

No importa el modo de error, si existe un fallo en la conexión siempre producirá una excepción, por eso siempre se conecta con try/catch.

- **PDO::ERRMODE_SILENT**. Es el **modo de error por defecto**. Si se deja así habrá que comprobar los errores de forma parecida a como se hace con **mysqli**. Se tendrían que emplear PDO::errorCode() y PDO::errorInfo() o su versión en PDOStatement PDOStatement::errorCode() y PDOStatement::errorInfo().
- **PDO::ERRMODE_WARNING**. Además de establecer el código de error, PDO emitirá un mensaje E_WARNING. Modo empleado para depurar o hacer pruebas para ver errores sin interrumpir el flujo de la aplicación.
- **PDO::ERRMODE_EXCEPTION**. Además de establecer el código de error, PDO lanzará una excepción PDOException y establecerá sus propiedades para luego poder reflejar el error y su información. Este modo se emplea en la mayoría de situaciones, ya que permite manejar los errores y a la vez esconder datos que podrían ayudar a alguien a atacar tu aplicación.

INSTANCIAR PDO

```
 */
public function __construct()
{
    try {
        $this->conexion = Conexion::conectar();
    } catch (Exception $e) {
        die($e->getMessage());
    }
}
```

FETCH ALL

```php
public function listar()
{
    try {
        $consulta = $this->conexion->prepare("SELECT * FROM categorias ORDER BY id ASC");
        $consulta->execute();

        return $consulta->fetchAll(PDO::FETCH_OBJ);
    } catch (Exception $e) {
        die($e->getMessage());
    }
}
```

ENCAPSULACIÓN

- **PDO::FETCH_ASSOC**: devuelve un array indexado cuyos keys son el **nombre de las columnas**.
- **PDO::FETCH_NUM**: devuelve un array indexado cuyos keys son **números**.
- **PDO::FETCH_BOTH**: valor por defecto. Devuelve un array indexado cuyos keys son tanto el **nombre de las columnas** como **números**.
- **PDO::FETCH_BOUND**: asigna los valores de las columnas a las variables establecidas con el método PDOStatement::bindColumn.
- **PDO::FETCH_CLASS**: asigna los valores de las columnas a propiedades de una clase. Creará las propiedades si éstas no existen.
- **PDO::FETCH_INTO**: actualiza una instancia existente de una clase.
- **PDO::FETCH_OBJ**: devuelve un objeto anónimo con nombres de propiedades que corresponden a las columnas.
- **PDO::FETCH_LAZY**: combina **PDO::FETCH_BOTH** y **PDO::FETCH_OBJ**, creando los nombres de las propiedades del objeto tal como se accedieron.

REGISTRAR

```php
public function registrar(Categoria $data)
{
    try {
        $insert = $this->conexion->prepare("INSERT INTO categorias (nombre) VALUES (?)");
        return $insert->execute(array($data->nombre));
    } catch (Exception $e) {
        die($e->getMessage());
    }
}
```

ACTUALIZAR

```php
public function actualizar(Categoria $data)
{
    try {
        $update = $this->conexion->prepare("UPDATE categorias SET nombre = ? WHERE id = ?");
        return $update->execute(array($data->nombre, $data->id));
    } catch (Exception $e) {
        die($e->getMessage());
    }
}
```

CONSULTAR

```php
public function obtener($id)
{
    try {
        $consulta = $this->conexion->prepare("SELECT * FROM categorias WHERE id = ?");
        $consulta->execute([$id]);
        return $consulta->fetch(PDO::FETCH_OBJ);
    } catch (Exception $e) {
        die($e->getMessage());
    }
}
```

BORRAR

```php
public function eliminar($id)
{
    try {
        $delete = $this->conexion->prepare("DELETE FROM categorias WHERE id = ?");
        return $delete->execute(array($id));
    } catch (Exception $e) {
        die($e->getMessage());
    }
}
```

BINDPARAM

Con bindParam() la variable es enlazada como una referencia y sólo será evaluada cuando se llame a execute():

```php
// Prepare:
$stmt = $this->conexion->prepare("INSERT INTO Clientes (nombre) VALUES (:nombre)");
$nombre = "Morgan";
// Bind
$stmt->bindParam(':nombre', $nombre); // Se enlaza a la variable $nombre
// Si ahora cambiamos el valor de $nombre:
$nombre = "John";
$stmt->execute(); // Se insertará el cliente con el nombre John
```

BINDVALUE

Con bindValue() se enlaza el valor de la variable y permanece hasta execute():

```php
// Prepare:
$stmt = $this->conexion->prepare("INSERT INTO Clientes (nombre) VALUES (:nombre)");
$nombre = "Morgan";
// Bind
$stmt->bindValue(':nombre', $nombre); // Se enlaza al valor Morgan
// Si ahora cambiamos el valor de $nombre:
$nombre = "John";
$stmt->execute(); // Se insertará el cliente con el nombre Morgan
```

TERCER PARÁMETRO

PDO::PARAM_BOOL (integer)
Representa un tipo de dato booleano.

PDO::PARAM_NULL (integer)
Representa el tipo de dato NULL de SQL.

PDO::PARAM_INT (integer)
Representa el tipo de dato INTEGER de SQL .

PDO::PARAM_STR (integer)
Representa el tipo de dato CHAR, VARCHAR de SQL, u otro tipo de datos de cadena.

EJEMPLO EN RETOS UPDATE

```
public function actualizar(Retos $data)
{
    try {
        $update = $this->conexion->prepare("UPDATE retos
        SET nombre = :nombre, dirigido = :dirigido, descripcion = :descripcion,
        fechaInicioInscripcion = :fechaInicioInscripcion,  fechaFinInscripcion = :fechaFinInscripcion,
        fechaInicioReto = :fechaInicioReto, fechaFinReto = :fechaFinReto,
        fechaPublicacion = :fechaPublicacion, publicado = :publicado,
        idCategoria = :idCategoria, idProfesor = :idProfesor
        WHERE id = :id");

        $update->bindValue(':nombre', $data->nombre);
        $update->bindValue(':dirigido', $data->dirigido);
        $update->bindValue(':descripcion', $data->descripcion);
        $update->bindValue(':fechaInicioInscripcion', $data->fii);
        $update->bindValue(':fechaFinInscripcion', $data->ffi);
        $update->bindValue(':fechaInicioReto', $data->fir);
        $update->bindValue(':fechaFinReto', $data->ffr);
        $update->bindValue(':fechaPublicacion', $data->fechaPublicacion);
        $update->bindValue(':publicado', $data->publicado, PDO::PARAM_INT); // aquí está la solución al Bit
        $update->bindValue(':idCategoria', $data->idCategoria);
        $update->bindValue(':idProfesor', $data->idProfesor);
        $update->bindValue(':id', $data->id);

        return $update->execute();
    } catch (Exception $e) {
        die($e->getMessage());
    }
```

TRANSACCIONES

```php
lic function altaGrupo($data)

try {
    $this->conexion->beginTransaction();
    $this->nombreGrupo = $data[0];
    if ($data[1] == '') {
        $tmp = NULL;
        $this->descripcion = $tmp;
    } else {
        $this->descripcion = $data[1];
    }
    $this->idClase = $data[2];
    $this->idReto = $data[3];
    $this->idProfesor = $data[4];

    $insert = $this->conexion->prepare('INSERT INTO grupos (nombre, descripcion, idClase, idReto, idProfesor) VALUES (?, ?, ?, ?, ?)');
    $resultado = $insert->execute(array($this->nombreGrupo, $this->descripcion, $this->idClase, $this->idReto, $this->idProfesor));
    if ($resultado) {
        $ultimo = $this->conexion->lastInsertId();
        $this->conexion->commit();
        return $ultimo;
    }
} catch (Exception $e) {
    $this->conexion->rollBack();
    // return $e->getCode();
}
```

TRATAMIENTO ERRORES

FUENTES

https://diego.com.es/tutorial-de-pdo

https://parzibyte.me/blog/2019/06/05/conectar-php-sql-server-pdo-crud-ejemplo/

https://www.php.net/manual/es/pdo.constants.php