PHP Data Objects (PDO)

Acceso a Bases de Datos en PHP

Desarrollo Web en Entorno Servidor

¿Qué es PDO?

PHP Data Objects (PDO) es una capa de abstracción para acceder a bases de datos en PHP.

- ✓ Interfaz común para múltiples SGBD
- ✓ Orientado a objetos (sin interfaz procedimental)
- ✓ Soporta: MySQL, PostgreSQL, SQLite, Oracle, SQL Server...
- ✓ Mayor portabilidad del código
- ✓ Consultas preparadas nativas
- ✓ Manejo robusto de excepciones

Ventajas de PDO

VENTAJAS

- Cambio fácil de SGBD
- Sintaxis consistente
- Seguridad mejorada
- Mejor rendimiento
- Manejo de errores robusto

PDO vs MySQLi

✓ Múltiples SGBD

✓ Solo POO

✓ Consultas nombradas

✓ Mayor portabilidad

✓ Estándar moderno

Establecer Conexión con PDO

Para conectar con una base de datos, se crea una instancia de la clase PDO:

```php
<?php
// Sintaxis básica
$pdo = new PDO('mysql:host=localhost;dbname=dwes',
               'usuario',
               'contraseña');

// Con opciones de configuración
$opciones = array(
    PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8"
);

$dwes = new PDO('mysql:host=localhost;dbname=dwes',
                'dwes',
                'abc123.',
                $opciones);
?>
```

Parámetros de Conexión (DSN)

Parámetro	Descripción	Ejemplo
host	Nombre o IP del servidor	localhost / 192.168.1.100
port	Puerto TCP del servidor	3306
dbname	Nombre de la base de datos	dwes
charset	Codificación de caracteres	utf8mb4
unix_socket	Socket Unix (solo Linux)	/tmp/mysql.sock

```
$dsn = "mysql:host=localhost;port=3306;dbname=dwes;charset=utf8mb4";
```

Ejecución de Consultas

1. Consultas de ACCIÓN (INSERT, UPDATE, DELETE) → método exec()

```
$registros = $dwes->exec('DELETE FROM stock WHERE unidades=0');
echo "Se eliminaron $registros registros";
```

2. Consultas de SELECCIÓN (SELECT) → método query()

```
$resultado = $dwes->query("SELECT producto, unidades FROM stock");

while ($row = $resultado->fetch()) {
    echo $row['producto'] . ": " . $row['unidades'] . "<br>";
}
```

Métodos fetch() para Obtener Resultados

Método	Descripción	Uso
PDO::FETCH_ASSOC	Array asociativo	$row['columna']
PDO::FETCH_NUM	Array numérico	$row[0]
PDO::FETCH_BOTH	Ambos (por defecto)	$row[0] o $row['columna']
PDO::FETCH_OBJ	Objeto	$row->columna
PDO::FETCH_BOUND	Asigna a variables	bindColumn()

```
// Usando FETCH_OBJ
while ($registro = $resultado->fetch(PDO::FETCH_OBJ)) {
    echo "Producto: " . $registro->producto . "<br>";
}
```

Método fetchAll() en PDO

fetchAll() recupera TODOS los registros del resultado en un solo array, en lugar de uno por uno.

✅ VENTAJAS

- Código más simple y corto
- Útil para datasets pequeños
- Permite recorrer varias veces
- Ideal con count(), empty()

⚠️ DESVENTAJAS

- Consume más memoria
- No recomendado para datasets grandes
- Carga todo en memoria de una vez

COMPARACIÓN: fetch() vs fetchAll()

```
// Con fetch() - uno por uno (eficiente en memoria)
$resultado = $pdo->query("SELECT * FROM productos");
while ($row = $resultado->fetch(PDO::FETCH_ASSOC)) {
    echo $row['nombre'] . "<br>";
}

// Con fetchAll() - todos de una vez (más simple)
$resultado = $pdo->query("SELECT * FROM productos");
$productos = $resultado->fetchAll(PDO::FETCH_ASSOC);

foreach ($productos as $producto) {
```

Consultas Preparadas

Las consultas preparadas mejoran la seguridad y el rendimiento

```
// Opción 1: Con signos de interrogación (?)
$consulta = $dwes->prepare('INSERT INTO familia (cod, nombre) VALUES (?, ?)');
$consulta->bindParam(1, $cod_producto);
$consulta->bindParam(2, $nombre_producto);
$consulta->execute();

// Opción 2: Con parámetros nombrados (:nombre)
$consulta = $dwes->prepare('INSERT INTO familia VALUES (:cod, :nombre)');
$consulta->bindParam(':cod', $cod_producto);
$consulta->bindParam(':nombre', $nombre_producto);
$consulta->execute();

// Opción 3: Pasar array en execute()
$consulta = $dwes->prepare('INSERT INTO familia VALUES (:cod, :nombre)');
$parametros = array(':cod' => 'TABLET', ':nombre' => 'Tablet PC');
$consulta->execute($parametros);
```

Transacciones en PDO

Las transacciones garantizan la integridad de los datos: todas las operaciones se ejecutan o ninguna

```php
<?php
try {
    // Iniciar transacción
    $dwes->beginTransaction();

    // Realizar operaciones
    $dwes->exec('DELETE FROM stock WHERE unidades=0');
    $dwes->exec('UPDATE stock SET unidades=3 WHERE producto="TABLET"');

    // Si todo va bien, confirmar cambios
    $dwes->commit();
    echo "Transacción completada exitosamente";

} catch (Exception $e) {
    // Si algo falla, revertir cambios
    $dwes->rollBack();
    echo "Error en transacción: " . $e->getMessage();
}
?>
```

Manejo de Excepciones

Configurar PDO para lanzar excepciones:

```php
<?php
// Activar modo de excepciones
$dwes->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

// Capturar excepciones
try {
    $sql = "SELECT * FROM tabla_inexistente";
    $resultado = $dwes->query($sql);

} catch (PDOException $e) {
    echo "Error: " . $e->getMessage() . "<br>";
    echo "Código: " . $e->getCode();
}

// Modos de error disponibles:
// - PDO::ERRMODE_SILENT     → No hace nada (por defecto)
// - PDO::ERRMODE_WARNING    → Genera E_WARNING
// - PDO::ERRMODE_EXCEPTION  → Lanza PDOException
?>
```

Buenas Prácticas con PDO

- **Seguridad** Siempre usar consultas preparadas con parámetros
- **Excepciones** Activar PDO::ERRMODE_EXCEPTION en producción
- **Credenciales** No hardcodear contraseñas, usar archivos de configuración
- **Fetch** Elegir el modo fetch apropiado según necesidad
- **Conexiones** Cerrar conexiones: $pdo = null;
- **Transacciones** Usar transacciones para operaciones relacionadas
- **Validación** Validar datos antes de insertar/actualizar
- **Depuración** Desactivar display_errors en producción

Ejemplo Completo: CRUD con PDO

```php
<?php
// 1. Conexión
$pdo = new PDO('mysql:host=localhost;dbname=dwes', 'dwes', 'abc123.');
$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

// 2. CREATE - Insertar
$sql = "INSERT INTO productos (nombre, precio) VALUES (:nombre, :precio)";
$stmt = $pdo->prepare($sql);
$stmt->execute([':nombre' => 'Laptop', ':precio' => 899.99]);

// 3. READ - Consultar
$sql = "SELECT * FROM productos WHERE precio < :precio";
$stmt = $pdo->prepare($sql);
$stmt->execute([':precio' => 1000]);
$productos = $stmt->fetchAll(PDO::FETCH_ASSOC);

// 4. UPDATE - Actualizar
$sql = "UPDATE productos SET precio = :precio WHERE id = :id";
$stmt = $pdo->prepare($sql);
$stmt->execute([':precio' => 799.99, ':id' => 1]);

// 5. DELETE - Eliminar
$sql = "DELETE FROM productos WHERE id = :id";
$stmt = $pdo->prepare($sql);
$stmt->execute([':id' => 1]);
?>
```