Acceso a Base de datos con PDO

Antonio Espín Herranz

Contenidos

- Acceso a BD con PDO
  - Conexiones.
  - Consultas parametrizadas
  - Implementación de un patrón DAO con las operaciones: CRUD, create, read, update y delete.
  - Gestión de transacciones
  - Gestión de errores

php.ini

- Comprobar si esta activada la directiva (descomentar):
- En D:\php8\php.ini (la ruta de instalación)
  - **extension=pdo_mysql**
  - **extension=mysqli**

  - **Puede que haya que tocar:**
  - **extension_dir = "D:\php8\ext“ con la ruta de la instalación**

Conexiones

```php
<?php
$host = 'localhost';
$dbname = 'mi_base_de_datos';
$user = 'usuario';
$password = 'clave';

try {
    $pdo = new PDO("mysql:host=$host;dbname=$dbname;charset=utf8mb4", $user, $password, [
        PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION, // lanza excepciones en errores
        PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC // devuelve arrays asociativos
    ]);
} catch (PDOException $e) {
    die("Error de conexión: " . $e->getMessage());
}
```

Conexiones

- La conexión y las propiedades de la configuración se puede hacer en dos pasos:

- php
- $pdo = new PDO("mysql:host=localhost;dbname=mi_bd", "usuario", "contraseña");
- Puedes configurar atributos como:

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

Consulta parametrizada

```
$categoria = 'Electrónica';

$sql = "SELECT nombre, precio FROM productos WHERE categoria = :categoria";
$stmt = $pdo->prepare($sql);
$stmt->execute(['categoria' => $categoria]);

$resultados = $stmt->fetchAll();

foreach ($resultados as $producto) {
   echo "{$producto['nombre']} - {$producto['precio']} €<br>";
}
```

// Evitan inyecciones de SQL

Patrón DAO

- Tomamos una clase abstracta que mantiene la conexión y permite realizar transacciones.

- Cualquier clase que implemente un dao deberá heredar de esta clase.

- Se puede definir una interface con las operaciones a implementar en la clase final.

Clase BaseDAO

```
namespace App\DAOs;
use PDO;
abstract class BaseDAO {
    protected PDO $conexion;

    public function __construct(PDO $conexion) {
        $this->conexion = $conexion;
    }
    public function comenzarTransaccion(): void {
        $this->conexion->beginTransaction();
    }
    public function confirmarTransaccion(): void {
        $this->conexion->commit();
    }
     public function revertirTransaccion(): void {
        $this->conexion->rollBack();
     }
}
```

Una clase abstracta no se puede instanciar directamente.

Dao – Create - insert

```
// Datos del producto
  $nombre = "Auriculares";
  $categoria = "Electrónica";
  $precio = 29.99;
  $existencias = 100;

  // Sentencia parametrizada
  $sql = "INSERT INTO productos (nombre, categoria, precio, existencias)
        VALUES (:nombre, :categoria, :precio, :existencias)";

  $stmt = $pdo->prepare($sql);
  $stmt->execute([
     'nombre' => $nombre,
     'categoria' => $categoria,
     'precio' => $precio,
     'existencias' => $existencias
  ]);
```

Dao – Update - Update

```
$sql = "UPDATE productos SET precio = :precio, existencias =
:existencias WHERE id = :id";
$stmt = $pdo->prepare($sql);
$stmt->execute([
    'precio' => $nuevoPrecio,
    'existencias' => $nuevasExistencias,
    'id' => $id
]);
```

Dao – Delete - Delete

```
$id = 2;


$sql = "DELETE FROM productos WHERE id = :id";
$stmt = $pdo->prepare($sql);
$stmt->execute(['id' => $id]);
```

Dao – Read - select

- Normalmente la carga de un objeto de la BD será por clave primaria. Puede que no la encuentre, en este caso de forma opcional puede devolver null, se indica con: ***?Producto (devuelve null o el objeto)***

```
public function read(int $id): ?Producto {
    $sql = 'SELECT * FROM montes WHERE id = :id';
    $stmt = $this->conexion->prepare($sql);
    $stmt->execute([':id' => $id]);
    $fila = $stmt->fetch(\PDO::FETCH_ASSOC); // Formato: Array asociativo
    return $fila ? new Producto(...) : null;
}
```

- En tipos int, float tendremos que forzar la conversión con casting:
  - $var = (int)$fila[‘id’];

Número de registros afectados

- En un insert, update o delete con PDO, para obtener el número de registros afectados:

```
$sql = "INSERT INTO productos (nombre, precio) VALUES (:nombre, :precio)";
$stmt = $pdo->prepare($sql);
$stmt->execute([
   ':nombre' => 'Camiseta',
   ':precio' => 19.99
]);

$filasAfectadas = $stmt->rowCount();
echo "Se insertaron $filasAfectadas fila(s).";
```

Ultimo id generado

- En un campo **autoincrement**, cuando insertamos un registro podemos obtener el id autogenerado por la BD:

```
$sql = "INSERT INTO productos (nombre, precio) VALUES (:nombre, :precio)";
$stmt = $pdo->prepare($sql);
$stmt->execute([
   ':nombre' => 'Camiseta',
   ':precio' => 19.99
]);

$ultimoId = $pdo->lastInsertId();
echo "El ID generado es: $ultimoId";
```

Transacciones

- No es necesario desactivar el modo autocommit de la BD:
- La llamada a **beginTransaction**() lo desactiva automáticamente.
- **commit**() → confirma transacción.
- **rollback**()→ Revertir la transacción.

Esquema de una transacción

```
$pdo = new PDO(...)
try {
    $pdo→beginTransaction();
    Ejecutar consultas de acción...
    $pdo→commit();

} catch (Exception $e){
    $pdo→rollBack();
}
```