Clase PDO – Lenguaje de programación PHP.

2º DAW

I.E.S. Los Sauces

Erika Martínez Pérez

Desarrollo web en entorno de servidor

índice de contenidos.

- Funcionalidad

La funcionalidad de la clase PDO (**PHP Data Object**) es la proporción de una interfaz para interactuar con base de datos desde aplicaciones realizadas con el lenguaje de programación PHP.

- Constantes

Modos de error:

a. **PDO::ERRMODE_WARNING**: Configura PDO para que emita advertencias.
b. **PDO::ERRMODE_EXCEPTION**: Configura PDO para que lance excepciones.

Atributos de conexión:

a. **PDO::ATTR_ERRMODE**: Atributo para configurar el modo de error.
b. **PDO::ATTR_DEFAULT_FETCH_MODE**: Atributo para configurar el modo de obtención de resultados predeterminado.

Tipos de obtención de resultados:

a. **PDO::FETCH_ASSOC**: Devuelve un array indexado por nombres de columnas.
b. **PDO::FETCH_OBJ**: Devuelve un objeto anónimo con nombres de propiedades que corresponden a los nombres de las columnas.

Modos de ejecución de consultas:

a. **PDO::FETCH_CLASS**: Instancia un objeto y lo carga con datos de la fila.

Atributos de declaración:

a. **PDO::ATTR_CASE**: Atributo para configurar la forma en que se manejan los nombres de columnas en los resultados.
b. **PDO::ATTR_CURSOR**: Atributo para configurar el tipo de cursor.
c. **PDO::ATTR_TIMEOUT**: Atributo para configurar el tiempo de espera de la conexión.

Otros:

a. **PDO::PARAM_***: Varios atributos para establecer el tipo de parámetro en las consultas preparadas (por ejemplo, PDO::PARAM_INT, PDO::PARAM_STR).

- Auto-Commit

El término auto-commit hace referencia a la capacidad de asegurar todas las transacciones de una base de datos, siendo una transaccion un conjunto de operaciones (select, update, delete, insert), dichas estas se realizan en su totalidad o por el contrario no se realizará ninguna.

La propiedad del auto-commit de PDO trata de manera independiente cada una de sus operaciones y realiza su confirmación automáticamente, si alguna de estas es fallida se realizará un rollback o vuelta atrás para no realizar ninguna de las operaciones.

Ejemplo:

```php
// Establecer la conexión a la base de datos
$dsn = "mysql:host=localhost;dbname=nombre_base_de_datos";
$usuario = "nombre_usuario";
$contrasena = "contrasena";

try {
    $conexion = new PDO($dsn, $usuario, $contrasena);
} catch (PDOException $e) {
    echo "Error de conexión: " . $e->getMessage();
    exit;
}

// Desactivar autocommit y comenzar una transacción
$conexion->setAttribute(PDO::ATTR_AUTOCOMMIT, 0);
$conexion->beginTransaction();

try {
    // Realizar operaciones que forman parte de la transacción
    $conexion->exec("UPDATE tabla SET columna = 'nuevo_valor' WHERE id = 1"

    // Confirmar la transacción
    $conexion->commit();
} catch (PDOException $e) {
    // Revertir la transacción en caso de error
    $conexion->rollBack();
    echo "Error: " . $e->getMessage();
}

// Restaurar autocommit a su estado predeterminado
$conexion->setAttribute(PDO::ATTR_AUTOCOMMIT, 1);
```

- Sentacias preparadas

Las sentencias preparadas son una técnica en programación de bases de datos que se utiliza para mejorar el rendimiento al ejecutar consultas SQL.

En el contexto de PHP y la clase PDO, una sentencia preparada es una plantilla de consulta SQL que se precompila y se almacena en el servidor de la base de datos antes de ejecutarla, pudiendo así darle diferentes valores a los parámetros de la consulta y ejecutarla tantas veces fuera necesaria.

Ejemplos:

```php
// Establecer la conexión a la base de datos
$dsn = "mysql:host=localhost;dbname=nombre_base_de_datos";
$usuario = "nombre_usuario";
$contrasena = "contrasena";

try {
    $conexion = new PDO($dsn, $usuario, $contrasena);
} catch (PDOException $e) {
    echo "Error de conexión: " . $e->getMessage();
    exit;
}

// Sentencia preparada
$consulta = $conexion->prepare("INSERT INTO tabla (columna1, columna2)

// Asignar valores a los parámetros
$valor1 = "ejemplo1";
$valor2 = "ejemplo2";

$consulta->bindParam(':valor1', $valor1);
$consulta->bindParam(':valor2', $valor2);

// Ejecutar la sentencia preparada
$consulta->execute();

// Puedes reutilizar la sentencia preparada con diferentes valores
$valor1 = "nuevo_ejemplo1";
$valor2 = "nuevo_ejemplo2";

$consulta->execute();
```

- Metodos

Conexión a la base de datos:

a. **__construct($dsn, $username, $password, $options)**: Constructor que establece una conexión a la base de datos. $dsn es el Data Source Name, y $options son opciones de configuración.

Preparación y ejecución de consultas:

a. **prepare($statement, $driver_options)**: Prepara una sentencia SQL para ser ejecutada por el motor de la base de datos.
b. **exec($statement)**: Ejecuta una sentencia SQL y devuelve el número de filas afectadas.
c. **query($statement)**: Ejecuta una sentencia SQL y devuelve un conjunto de resultados como un objeto PDOStatement.

Manipulación de resultados:

a. **fetch($fetch_style, $cursor_orientation, $cursor_offset)**: Recupera la siguiente fila de un conjunto de resultados.
b. **fetchAll($fetch_style, $fetch_argument, $ctor_args)**: Devuelve un array que contiene todas las filas del conjunto de resultados.

Transacciones:

a. **beginTransaction()**: Inicia una transacción.
b. **commit()**: Confirma la transacción actual.
c. **rollBack()**: Revierte la transacción actual.

Gestión de errores:

a. **errorCode()**: Devuelve el código de error de la última operación de la base de datos.
b. **errorInfo()**: Devuelve información extendida sobre los errores ocurridos durante la última operación de la base de datos.

Configuración y ajustes:

a. **setAttribute($attribute, $value)**: Establece atributos de conexión.
b. **setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION)**: Configura PDO para que lance excepciones en caso de errores.