

Alcantara Huerta Angel Josue



**Escuela:** Universidad Tecnológica de Tijuana

**Carrera:** TI Desarrollo de Software Multiplataforma

**Asignatura:** C-Desarrollo de aplicaciones Web

**Trabajo:** PHP y MySQL

**Alumno:** Alcantara Huerta Angel Josue

**Docente:** Ray Brunnet Parra Galaviz

**Fecha de entrega:** 18 de octubre de 2024

**Grupo:** 3C

## Diferentes Cuales son las maneras de conectar PHP a MySQL

Existen varias formas de conectar PHP a una base de datos, dependiendo de las características y flexibilidad que busques. Las más comunes son MySQLi, PDO, y la ya obsoleta función MySQL.

### 1. MySQLi (MySQL Improved)

MySQLi es una extensión mejorada para interactuar con bases de datos MySQL. Ofrece una interfaz orientada a objetos, pero también puede usarse con un estilo procedural. Las ventajas de MySQLi incluyen soporte para consultas preparadas, lo que ayuda a prevenir ataques de inyección SQL.

Características principales:

- Soporte para consultas preparadas.
- Modo orientado a objetos y procedural.
- Específica para MySQL (no funciona con otras bases de datos).

### 2. PDO (PHP Data Objects)

PDO es una interfaz más genérica que soporta múltiples sistemas de bases de datos, no solo MySQL, lo que la hace más flexible si planeas migrar a otra base de datos en el futuro (como PostgreSQL o SQLite). PDO también ofrece un manejo de errores más robusto y un mejor soporte para consultas preparadas.

Características principales:

- Soporta múltiples bases de datos (MySQL, PostgreSQL, SQLite, entre otras).
- Soporte completo para consultas preparadas, que ayuda a evitar inyecciones SQL.
- Manejo de errores mediante excepciones.
- Menos funciones específicas de MySQL comparado con MySQLi.

### 3. Extensión MySQL (Obsoleta)

La extensión MySQL era la forma original de conectar PHP a bases de datos MySQL, pero ha sido eliminada desde PHP 7.0. Se considera insegura porque no soporta consultas preparadas, lo que la hacía vulnerable a ataques como la inyección SQL. No debe usarse en proyectos nuevos.

## Como hacer estas conexiones?

### MySQLi

Crear variables que serán pasadas posteriormente como parámetros al método `mysqli_connect()`. Los cuatro parámetros necesarios que tenemos que pasar la función son: `$servername` (dirección del servidor), `$database` (base de datos), `$username` (usuario) y `$password` (contraseña).

Cuando se realiza la conexión la base de datos, si no es correcta, se ejecuta la función `die()`. Lo que hace esta función es detener la ejecución por completo y mostrar un mensaje de error y/o advertencia seguido de un mensaje del error exacto que describe el problema.

Si la conexión se realiza de forma correcta, se mostrará un mensaje de "Conexión satisfactoria".

Por último, se cerrará la conexión con la base de datos utilizando la función `mysqli_close()`, a la cual le pasaremos como variable la conexión creada previamente. Si no especificamos el cierre de la conexión, MySQL cerrará por sí solo la conexión una vez que finalice el script.

A screenshot of a code editor with a dark background and light-colored text. The code is a PHP script for connecting to a MySQL database using MySQLi. It includes comments in Spanish for each line. The code is as follows:

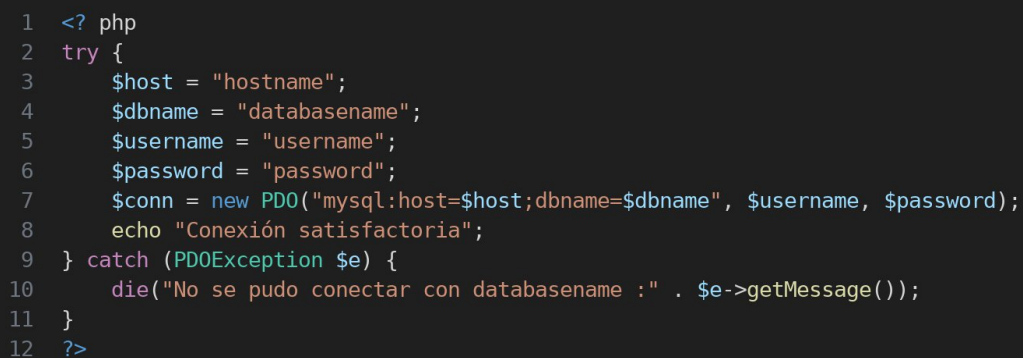
```
1  <? php
2  $servername = "servername"; // Nombre/IP del servidor
3  $database = "databasename"; // Nombre de la BBDD
4  $username = "username"; // Nombre del usuario
5  $password = "password"; // Contraseña del usuario
6  // Creamos la conexión
7  $con = mysqli_connect($servername, $username, $password, $database);
8  // Comprobamos la conexión
9  if (!$con) {
10     die("La conexión ha fallado: " . mysqli_connect_error());
11 }
12 echo "Conexión satisfactoria";
13 mysqli_close($con);
14 ?>
```

## PDO

Cuando creamos una conexión a una base de datos MySQL utilizando PDO, necesitamos crear un nuevo objeto PDO con un: (DSN – Data Source Name), nombre de usuario y contraseña.

Deberemos definir un DSN que es el tipo de base de datos, un \$host (nombre o ip del servidor), un \$dbname (nombre de la base de datos), un \$username (nombre de usuario) y un \$password (contraseña). Estas variables las declararemos antes de realizar la conexión.

En este ejemplo, utilizaremos un try/catch que básicamente significa que ejecutará lo que hay dentro del "try" y si hay un error ejecutará la parte del "catch". Si la conexión es correcta, se mostrará el mensaje "Conexión satisfactoria". Si la conexión resulta fallida, se mostrará un mensaje de error y se detendrá la ejecución.



```
1  <? php
2  try {
3      $host = "hostname";
4      $dbname = "databasename";
5      $username = "username";
6      $password = "password";
7      $conn = new PDO("mysql:host=$host;dbname=$dbname", $username, $password);
8      echo "Conexión satisfactoria";
9  } catch (PDOException $e) {
10     die("No se pudo conectar con databasename : " . $e->getMessage());
11 }
12 ?>
```

## MySQL(Obsoleta)

`mysql_connect()`: Se usaba para establecer la conexión con un servidor MySQL.

Parámetros:

- El host (normalmente "localhost").
- El nombre de usuario de la base de datos.
- La contraseña correspondiente al usuario.

`mysql_select_db()`: Elegía la base de datos que se iba a utilizar.

Parámetros:

- El nombre de la base de datos.
- La conexión, que era la conexión devuelta por `mysql_connect()`.

Verificación de errores: Si ocurría un error en la conexión o al seleccionar la base de datos, se usaba la función `mysql_error()` para obtener información del fallo.

Ejecución de consultas: Se usaba `mysql_query()` para ejecutar una consulta SQL.

Cerrar la conexión: Se podía cerrar explícitamente la conexión con `mysql_close()`, aunque no era estrictamente necesario, ya que PHP lo hacía automáticamente al finalizar el script.

```
1  <?php
2  $connection = mysql_connect("localhost", "username", "password");
3  if (!$connection) {
4      die('Could not connect: ' . mysql_error());
5  }
6
7  mysql_select_db("database_name", $connection);
8  mysql_close($connection);
9  ?>
```