

# Conexión con bases de datos en PHP

MySQLi & PDO

# MySQLi y PDO

Existen dos vías para acceder a BBDD en PHP:

- Extension MySQLi (**MySQL Improved**): solo para MySQL
- Extension PDO (**PHP Data Objects**): funciona con distintas BBDD.

MySQLi viene instalado por defecto.

Detalles de instalación de MySQLi:

<http://php.net/manual/en/mysqli.installation.php>

Detalles de instalación de PDO:

<http://php.net/manual/en/pdo.installation.php>



# MySQLi y PDO

```
// mysqli, procedural way
```

```
$conn = mysqli_connect('localhost','username','password','database');
```

```
// mysqli, object-oriented way
```

```
$conn = new mysqli('localhost','username','password','database');
```

```
// PDO, object-oriented way
```

```
$conn = new PDO("mysql:host=localhost;dbname=database", 'username', 'password');
```



# Función mysqli\_report()

*mysqli\_report()* habilita a MySQLi a lanzar excepciones. Se suele emplear con los parámetros **MYSQLI\_REPORT\_ERROR** y **MYSQLI\_REPORT\_STRICT**.

- MYSQLI\_REPORT\_ERROR indica a PHP que active los errores
- MYSQLI\_REPORT\_STRICT indica a PHP que convierta estos errores a excepciones

```
<?php
...
// antes de abrir la BBDD
mysqli_report(MYSQLI_REPORT_ERROR | MYSQLI_REPORT_STRICT);
...
?>
```

# Esquema de conexión y ejecución de comandos SQL

## 1. FASE DE **CONEXIÓN**

- Abrir la BBDD

## 2. FASE DE **EJECUCIÓN**

- Petición/consulta (Query) a la BBDD
- Fetch del query (solo si es necesario)

## 3. FASE DE **DESCONEXION**

- Cerrar la BBDD



# Establecer conexión con la BBDD - mysqli

```
DEFINE('HOST','localhost');
DEFINE('DBNAME','HR');
DEFINE('USERNAME','HR');
DEFINE('PASSWD','Educacio123!');
$conn = null;

// Configurar MySQLi para enviar Exceptions
mysqli_report(MYSQLI_REPORT_ERROR | MYSQLI_REPORT_STRICT);
try {
    $conn = new mysqli(HOST, USERNAME, PASSWD, DBNAME); // FASE DE CONEXION

    // FASE DE EJECUCION

    $conn->close(); // FASE DE DESCONEXION
}
catch (mysqli_sql_exception $e) {
    echo "Error connecting to MySQL: " . $e->getMessage();
}
```

# Fase de Ejecución - Ejemplo 1- Consulta

```
// FASE DE EJECUCION
```

```
$query = 'SELECT last_name, first_name FROM employees';
```

```
$table = $conn->query($query);
```

```
// query de DDBB
```

```
foreach ($table as $row) {
```

```
// fetch de DDBB
```

```
    foreach ($row as $column) {
```

```
        echo $column . ' ';
```

```
    }
```

```
}
```

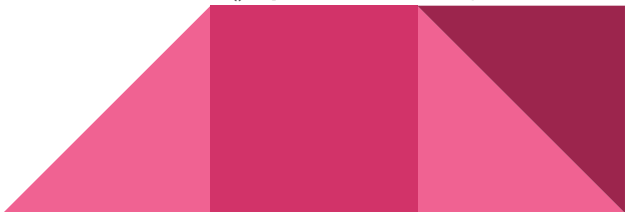


## Fase de Ejecución - Ejemplo 2 - Consulta

```
// FASE DE EJECUCION
$query = 'SELECT employee_id, last_name, first_name FROM employees ';
$table = $conn->query($query); // query the database
while( $row = $table->fetch_assoc() ) { // fetch the database
    echo $row['employee_id'] ." " . $row['last_name'] ." "
        . $row['first_name'] ."<br>";
}
```

`$table->fetch_assoc()` devuelve un array asociativo con strings representativas de la fila de la query resultante. Devuelve NULL si ya no quedan más filas en el resultado.

También se puede emplear `$table->fetch_array()` en lugar de `$table->fetch_assoc()`, pero el array se tendrá que tratar mediante claves numéricas.





# Fase de Ejecución - Ejemplo 2 - Inserción

```
$DEPARTMENT_NAME = "TEST";  
$DEPARTMENT_ID = 999;  
$query = "INSERT INTO departments( department_id, department_name )  
VALUES( " . $DEPARTMENT_ID . ", ' " . $DEPARTMENT_NAME . "' )";
```

```
INSERT INTO departments( department_id, department_name )  
VALUES( 999, 'TEST' )
```

```
$table = $conn->query($query); // insert en BBDD
```



# Consultas preparadas

- Previenen inyecciones SQL.
- Separan datos de la lógica de consulta.


Se consigue mediante:

- **prepare()**
- **bind\_param()** →
- **execute()**

Carácter	Descripción
i	la variable correspondiente es de tipo entero
d	la variable correspondiente es de tipo double
s	la variable correspondiente es de tipo string
b	la variable correspondiente es un blob (gran objeto binario) y se envía en paquetes

## Ejemplo - Ejecutar querys “preparadas”

```
$stmt = $mysqli->prepare("SELECT * FROM usuarios WHERE correo = ? AND id = ?");  
$correo = "ejemplo@correo.com";      // tipo string  
$id = 59;                             // tipo integer  
$stmt->bind_param("si", $correo, $id);  
$stmt->execute();  
$resultado = $stmt->get_result();  
while ( $fila = $resultado->fetch_assoc()) {  
    echo "Nombre: " . $fila["nombre"] . "<br>";  
}
```

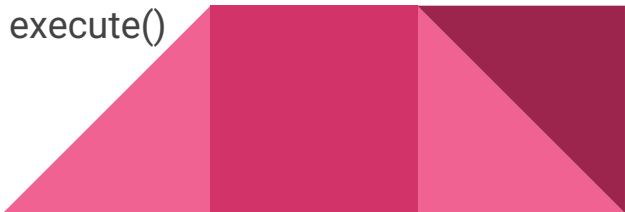


# Query() y Execute() con mysqli

- `mysqli::query` - Lanza una consulta sobre la BBDD. NO ofrece protección contra inyección de código SQL.
- `mysqli_stmt::execute` - Ejecuta la consulta preparada sobre la BBDD.

Con `execute`, al lanzar una consulta preparada, no existe riesgo de inyección de código SQL, por lo que es la opción recomendada, sobretodo al trabajar con consultas con variables (pe. `POST[ ]`).

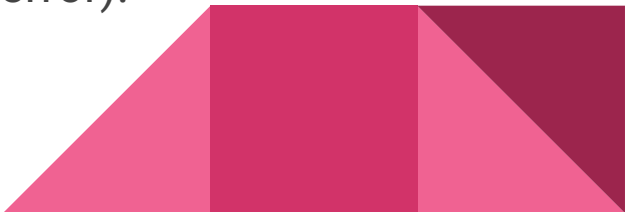
Funciones para trabajar con los resultados de un `execute`:

- `fetch_assoc()` - Devuelve los datos fila a fila, en formato array con claves como nombre de columnas. Para `query()` y `execute()`
  - `bind_result()` y `fetch()` - Devuelve los datos fila a fila, asignando valores a las variables indicadas en `bind_result()`. Solo para `execute()`
  - `get_result()` - Devuelve todo el resultado como un objeto. Solo para `execute()`
- 

# Transacciones


Permiten agrupar consultas y revertir cambios en caso de error.

Se consigue mediante:

- **begin\_transaction()** - Inicia una transacción
  - **commit()** - Aplica permanentemente los cambios hechos dentro de una transacción.
  - **rollback()** - Permite revertir los cambios (en caso de error).  
NO deshace un commit.
- 

# Ejemplo - Ejecutar querys como transacción

```
$mysqli->begin_transaction();  
try {  
    $mysqli->query("INSERT INTO usuarios (nombre, id) VALUES ('Juan', 1000)");  
    $mysqli->query("INSERT INTO usuarios (nombre, id) VALUES ('Maria', 2000)");  
    $mysqli->commit();  
} catch (Exception $e) {  
    $mysqli->rollback();  
    echo "Error: " . $e->getMessage();  
}
```



# Transacciones preparadas (todo junto)


```
$mysqli->begin_transaction();  
try {  
    $stmt = $mysqli->prepare("INSERT INTO usuarios (nombre, id)  
                             VALUES ('Juan', 1000)");  
  
    $stmt->execute();  
    $mysqli->commit();  
} catch (Exception $e) {  
    $mysqli->rollback();  
    echo "Error: " . $e->getMessage();  
}
```



# Autocommit()

En caso de que no queramos estar haciendo “commit” repetidas veces.

```
$mysqli->autocommit(true);
$mysqli->begin_transaction();
try {
    $stmt = $mysqli->prepare("INSERT INTO usuarios (nombre, id)
                                VALUES ('Juan', 1000)");
    $stmt->execute(); // Se realiza commit automaticamente
    // $mysqli->commit(); // esta sentencia ya no es necesaria
} catch (Exception $e) {
    $mysqli->rollback();
    echo "Error: " . $e->getMessage();
}
```





# PDO

- **PHP Data Objects.**
- Una interfaz unificada (vía PDO) para acceder a diferentes bases de datos.
- Compatible con **múltiples** sistemas de gestión de **bases de datos**, incluyendo MySQL, PostgreSQL, SQLite, entre otros.
- **Fase de cierre automática:** PHP cierra la conexión cuando el objeto se destruye o cuando el script termina.

Para establecer conexión:

```
$pdo = new PDO("mysql:host=localhost;dbname=mi_base_datos;charset=utf8mb4",  
               $usuario, $contraseña);
```



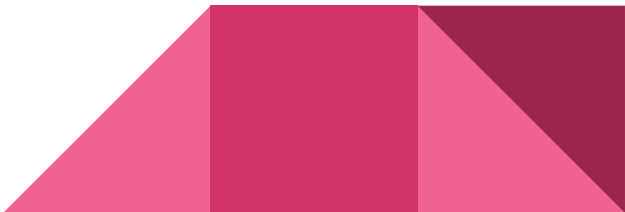
# Fase de conexión con PDO

```
$dsn = 'mysql:host=localhost;dbname=mi_base_datos;charset=utf8mb4';
```

```
$usuario = 'usuario';
```

```
$contraseña = 'contraseña';
```

```
try {  
    $pdo = new PDO($dsn, $usuario, $contraseña);  
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);  
    echo "Conexión establecida.";  
} catch (PDOException $e) {  
    echo "Error en la conexión: " . $e->getMessage();  
}
```




# Fase de Ejecución en PDO - Consultas preparadas

```
$sql = "SELECT * FROM usuarios WHERE correo = :correo";  
$stmt = $pdo->prepare($sql);  
$stmt->bindParam(':correo', $correo, PDO::PARAM_STR);  
$correo = 'ejemplo@correo.com';  
$stmt->execute();  
$resultado = $stmt->fetchAll(PDO::FETCH_ASSOC);  
print_r($resultado);
```




# Fase de Ejecución en PDO - Transacción

```
try {  
    $pdo->beginTransaction();  
    $pdo->exec("INSERT INTO usuario (nombre, id) VALUES ('Juan', 1000)");  
    $pdo->exec("INSERT INTO usuario (nombre, id) VALUES ('Maria', 2000)");  
    $pdo->commit();  
}  
catch (Exception $e) {  
    $pdo->rollBack();  
    echo "Fallo: " . $e->getMessage();  
}
```



# Fase de Ejecución en PDO - Transacciones preparadas

```
try {  
    $pdo->beginTransaction();  
    $stmt = $pdo->prepare("INSERT INTO usuario (nombre, id) VALUES ('Juan', 1000)");  
    $stmt->execute();  
    $pdo->commit();  
}  
catch (Exception $e) {  
    $pdo->rollBack();  
    echo "Fallo: " . $e->getMessage();  
}
```



# Query(), Exec() y Execute() en PDO

- PDO::**exec** - Para lanzar una consulta SQL no preparada y que no devuelve resultados
- PDO::**query** - Para lanzar una consulta SQL no preparada, pero que sí devuelve resultados
- PDOStatement::**execute** - Para lanzar una consulta SQL preparada, independientemente de si devuelve resultados o no

**Siempre** es recomendable **preparar las consultas** mediante prepare() y execute(). En especial si contienen input variable (pe. valores traídos mediante POST[ ]).

