




PHP

PDO - Clase 1



¿Cuál es el objetivo de hoy?

Ejecutar consultas en nuestra base de datos y obtener información.



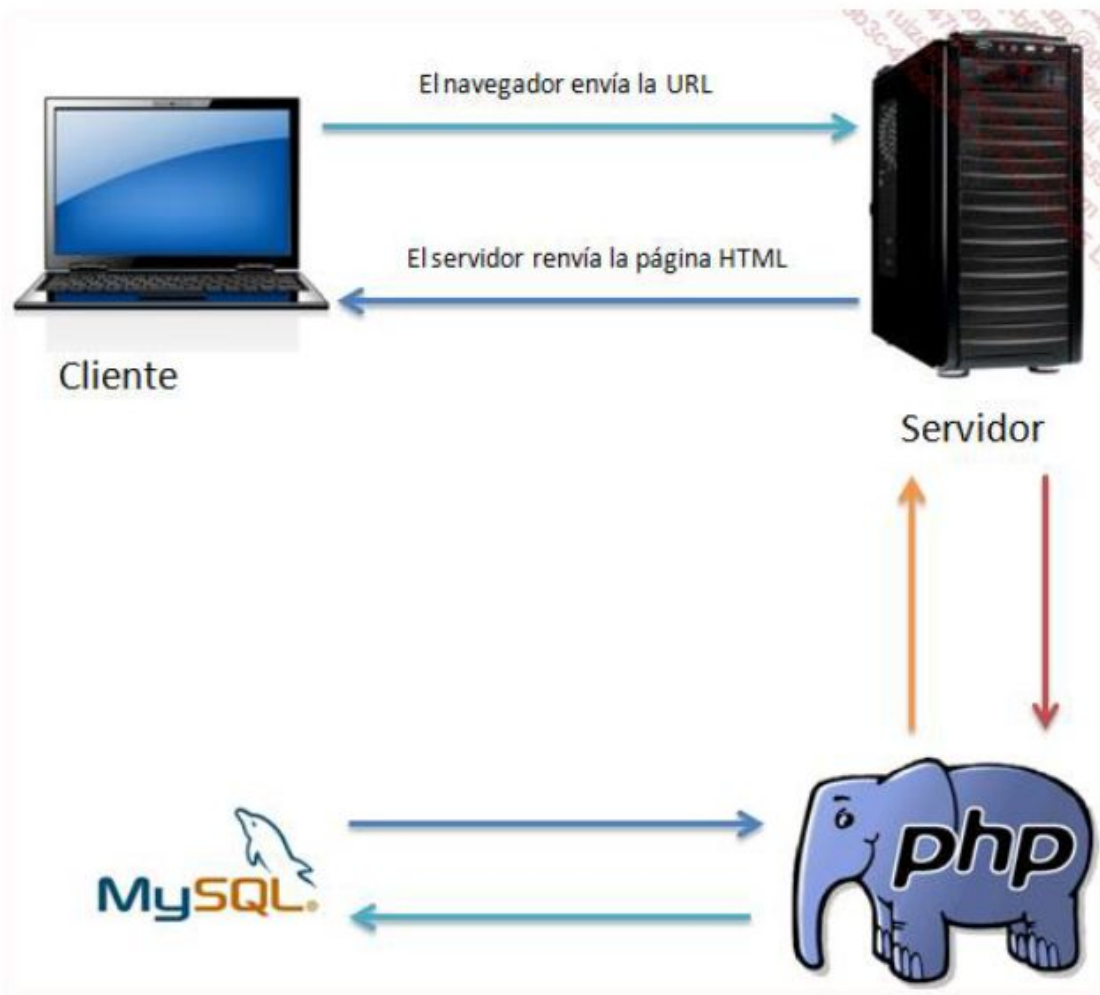


**Conectar nuestra app en PHP a la
base de datos, ¿por qué?**



¿Cómo lo hacemos?







Hace mucho tiempo...

```
<?php
```

```
    $link = mysql_connect('127.0.0.1', 'user', 'pass');
```

```
    mysql_select_db('testdb', $link);
```

```
    mysql_set_charset('UTF-8', $link);
```

```
    $query = mysql_query('Select * from peliculas', $link);
```

```
    $results = mysql_fetch_rows($query);
```

```
...
```



Todavía...

```
<?php
    $mysqli = new mysqli('127.0.0.1', 'usuario_db',
'password_db', 'nombre_db');

if (mysqli_connect_error()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}
```



¡Ahora!

```
<?php
$db = new PDO(
    'mysql:host=127.0.0.1;dbname=testdb;port=3306',
    'username',
    'password'
);
```


Sintaxis

Esta es la sintaxis que vamos a utilizar:

New PDO



¡Notación especial de objetos!

Por el momento no nos preocupemos, en las próximas clases vamos a comprender cómo utilizarlo.

PDO

PHP Data Object



Define una interfaz ligera para poder acceder a cualquier base de datos en PHP. PDO proporciona una capa de abstracción de acceso a datos.





Ventajas

- ◇ Múltiples Manejadores de Bases de Datos.
- ◇ Soporta últimas actualizaciones SQL:
 - **Reutilizar sentencias.**
 - **Transacciones.**
- ◇ Interfaz más amigable.
- ◇ **Parámetros** para hacer queries optimizados.
- ◇ Más funciones y **Control de errores.**
- ◇ Viene con PHP (últimas versiones).

Abrir una conexión

```
<?php
    $dsn =
'mysql:host=127.0.0.1;dbname=movies_db;port=3306';
    $db_user = 'root';
    $db_pass = '123456';
    $opt = [ PDO::ATTR_ERRMODE
              => PDO::ERRMODE_EXCEPTION ];
    $db = new PDO($dsn, $db_user, $db_pass, $opt);
```

dsn: Data Source Name

Cerrar una conexión

```
<?php
    $dsn = 'mysql:host=127.0.0.1;dbname=movies_db;port=3306';
    $db_user = 'root';
    $db_pass = '123456';
    $opt = [ PDO::ATTR_ERRMODE
              => PDO::ERRMODE_EXCEPTION ];
    $db = new PDO($dsn, $db_user, $db_pass, $opt);
```

`$db = null; //Aunque no es necesario`



Si hay errores
¿qué hacemos?





Capturarlos usando:

`if(){ } else { }`

`try{ } catch(){ }` [Gestion de Errores de PHP](#)



Capturar excepciones

```
<?php
    try {
        $db = new PDO($host, $db_user, $db_pass, $opt);
    }
    catch( PDOException $Exception ) {
        echo $Exception->getMessage();
    }
?>
```

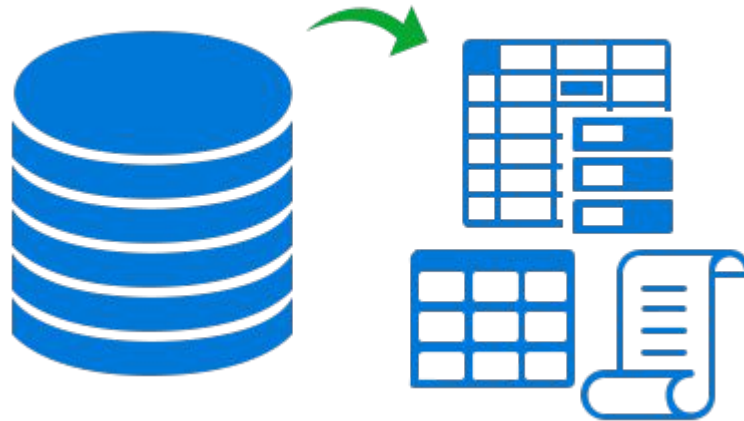

¡A practicar!

Conectarse a MySQL



```
<?php  
    new PDO( ... );  
?>
```

Ahora a obtener datos



Query sencillo (pasos)





El método/función: Query

Ejecuta una sentencia SQL, devolviendo un conjunto de resultados como un “objeto” PDOStatement.



Ejecutar

```
<?php
    try{
        ...$db = new PDO($host, $db_user, $db_pass, $opt);
    }catch...
```

```
$query = $db->query('SELECT * from peliculas');
```

¿Qué obtenemos?

PDO utiliza Objetos



Ejecutar

```
<?php
    try{
        ...$db = new PDO($host, $db_user, $db_pass, opt);
    }catch...
```

\$query = \$db->query('SELECT * from peliculas');

Trás haber ejecutado **new PDO** utilizaremos dicha variable de retorno para ejecutar métodos a base de datos.

Dichos métodos se ejecutan con el operador flecha ->

Ejecutar

```
<?php
$opt= [PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION];
$db = new PDO($host, $db_user, $db_pass, $opt);
try{
    $query = $db->query('SELECT * from peliculas');
}catch( PDOException $Exception ) {
    echo $Exception->getMessage();
}
```

Podemos también capturar **excepciones** cuando el nombre de la tabla o campos no existan, por eso usamos PDO::ERRMODE_EXCEPTION en las opciones.

Obtener

```
<?php
$opt= [PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION];
$db = new PDO($host, $db_user, $db_pass, $opt);
try{
    $query = $db->query('SELECT * from peliculas');
} catch( PDOException $Exception ) {
    echo $Exception->getMessage();
}
```

```
$results = $query->fetchAll(PDO::FETCH_ASSOC);
```



Obtener

Métodos

- `fetch` (<http://php.net/manual/en/pdostatement.fetch.php>)
- `fetchAll` (<http://php.net/manual/en/pdostatement.fetchall.php>)
- `rowCount` (<http://php.net/manual/en/pdostatement.rowcount.php>)

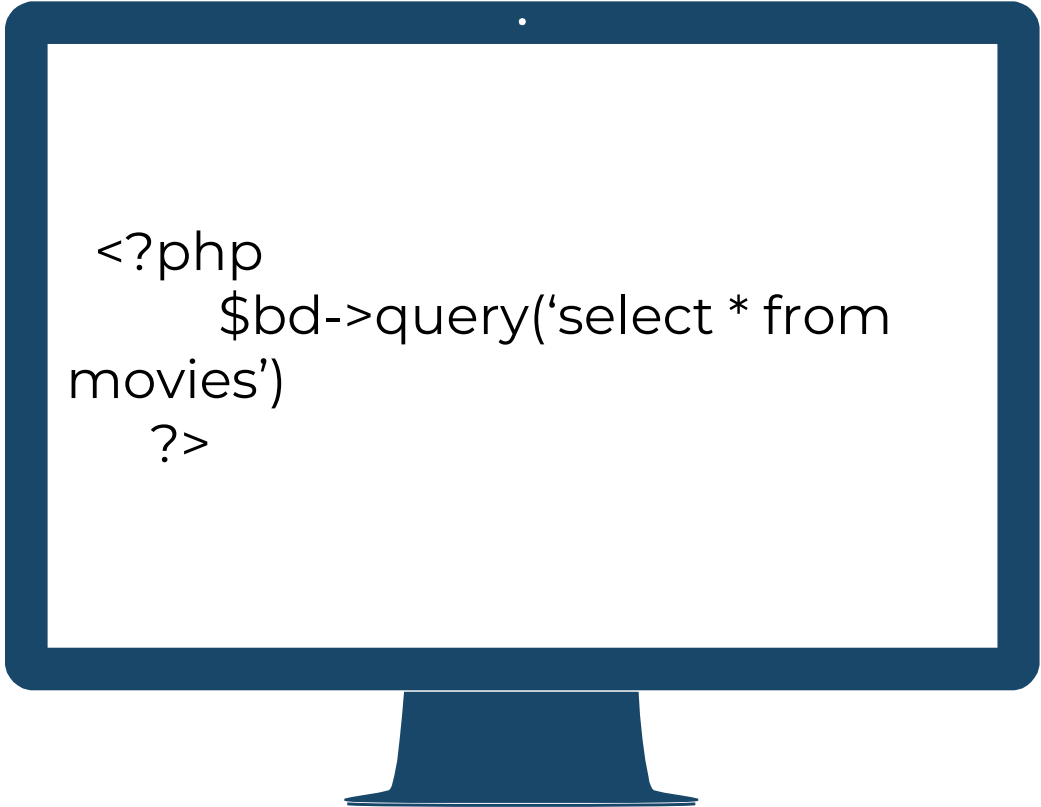
Constantes

(<http://php.net/manual/en/pdo.constants.php>)

- `PDO::FETCH_ASSOC`

¡A practicar!

Prácticas con queries



```
<?php
    $bd->query('select * from
movies')
?>
```



Statements

Plantillas compiladas para SQL.

- La consulta sólo necesita ser analizada (o preparada) **una** vez, pero puede ser ejecutada muchas veces con los mismos o diferentes parámetros.

[Documentacion](#)



Statements (pasos)





Definir

```
<?php
```

```
$db = new PDO($host, $db_user, $db_pass, $opt);
```

```
$query = $db->prepare('SELECT * from peliculas');
```



Ejecutar

```
<?php
    $db = new PDO($host, $db_user, $db_pass, $opt);

    $query = $db->prepare('SELECT * from peliculas');

    $query->execute();
```



Obtener

```
<?php
    $db = new PDO($host, $db_user, $db_pass, $opt);

    $query = $db->prepare('SELECT * from peliculas');

    $query->execute();

    // obtenemos los datos
    $result = $query->fetchAll(PDO::FETCH_ASSOC);
```


INSERT

```
<?php
```

```
    $db = new PDO($host, $db_user, $db_pass);
```

```
    $sql = "INSERT INTO peliculas (id, titulo, anio)  
VALUES (102, 'Big Fish', 2013)";
```

```
    $query = $db->prepare($sql);
```

```
    $query->execute(); //primera insercción
```

```
    $query->execute(); //segunda insercción??
```

```
    $query->execute(); //tercera insercción???
```

```
    $db = null;
```

```
// aplica lo mismo para update y delete. Inclusive para  
crear o alterar las BD.
```

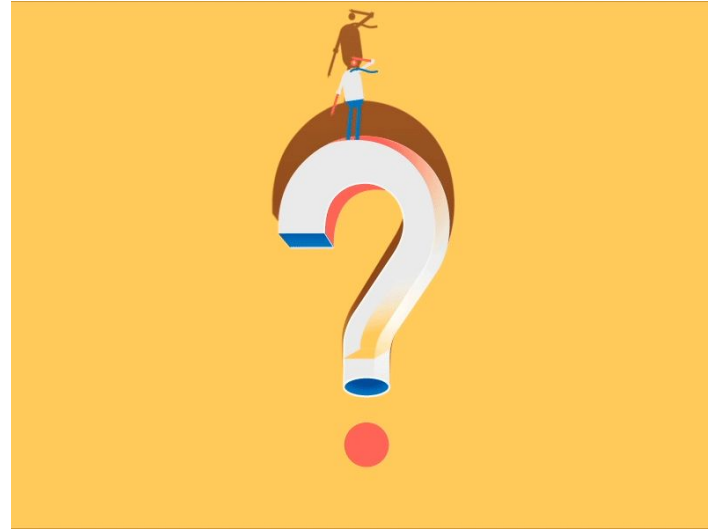
¡A practicar!

Statements



```
<?php  
    echo "Hora de practicar!";  
?>
```

¡Gracias!



¿Preguntas?