# PHP MySQLi

# y

# PDO

Antonio Boronat Pérez

IES Joan Coromines (Benicarló)

# Funciones de PHP con MySQL.

Una vez que conocemos las características básicas de MySQL y las sentencias SQL que se pueden ejecutar, vamos a ver como se integra MySQL con PHP.

Las definiciones de bases de datos y tablas las crearemos desde la consola de MySQL, preferiblemente a través de un script SQL con las sentencias de creación e inicialización de tablas, o mejor mediante algún asistente  web como PHPMyAdmin. La manipulación de datos en cambio se realizará desde PHP. Por lo tanto, las operaciones que nos interesan son:

• Leer datos.

• Añadir datos.

• Modificar datos.

• Eliminar datos.

PHP accede a MySQL a través de API, hay tres, pero PHP7 soporta MySQLi y PDO. Dispone de una extensión o biblioteca con múltiples funciones, que nos permiten acceder a la base de datos.

## Conectar con la base de datos.

Antes de empezar a consultar información tenemos que conectar con el sistema gestor de bases de datos y declarar la base de datos que vamos a usar. Esto se deberá hacer en cualquier aplicación que posteriormente quiera leer, añadir, modificar o eliminar datos.

### mysqli_connect

Esta función establece una conexión con la base de datos devolviendo un objeto de la clase *mysqli* que usaremos para nuestra sesión, si se procude un error devuelve el valor *false*:

$*conexion_bd = @*[1]*mysqli_connect(Servidor, Usuario, Contraseña, [Base de datos], [Puerto]);*

Los parámetros que recibe son:

- *Servidor*: nombre o dirección IP del servidor donde está alojada la base de datos. Si está instalado en la misma máquina que el servidor web, lo normal es que sea *localhost*. Se

1 @ es opcional, suprime los mensajes de error por si queremos tratarlos de otra manera.

puede elegir un número de puerto diferente del que se usa por defecto indicándolo en la misma función.

- *Usuario*: nombre de usuario con privilegios para acceder a la base de datos.
- *Contraseña*: contraseña del usuario.
- *Base de datos* indicaría, en caso de ponerla, la base de datos dentro del SGDB con la que vamos a trabajar. Si no se establece en la conexión se seleccionará como se muestra más adelante.
- *Puerto*, el puerto en que atiende el servidor si no es el definido por defecto.

La función *mysqli_connect()* está sobrecargada de forma que podemos elegir los parámetros que le pasamos en la llamada, de hecho si la ejecutamos sin ningún parámetro establecerá la conexión según una configuración por defecto de PHP, que se establece en el fichero */etc/php/7.0/apache2/php.ini*.

## mysqli_close

Esta función cierra una conexión con la base de datos, devolviendo un valor booleano que expresa si el cierre se ha producido correctamente o no.

*$cierre_bd = @mysqli_close($conexion_bd);*

i*f(!$cierre_bd)*

    *echo “Error en el cierre de la conexión a la BD.”;*

## Persistencia de la conexión a la BD

El funcionamiento normal de las conexiones a bases de datos es que al terminar la ejecución del programa se cierra la conexión, en caso de no haberla cerrado explícitamente con la función anterior, pero el proceso de apertura de una conexión a la base de datos y su cierre suponen una carga de procesamiento que puede relentizar la respuesta de la aplicación, así que es deseable disponer de conexiones persistentes, de forma que se mantenga operativas entre ejeciones de la aplicación sin la necesidad de abrirla cada vez mejorando el rendimiento. Para conseguir esto en MySQLi debemos poner **p:** delante del parámetro del servidor.

$*conexion_bd = @mysqli_connect(**p:Servidor**, Usuario, Contraseña, [Base de datos], [Puerto]);*

Debe estar configurado *php.ini* con los valores apropidados en [MySQLi], por defecto está disponible.

## mysqli_select_db

Con *mysqli_select_db* podemos seleccionar la base de datos si no la indicamos en la operación de conexión o bien la podemos cambiar para usar otra. Esta función devuelve un valor booleano que indica si la operación fue correcta.

*$seleccion_bd = mysqli_select_db($conexion_bd, Base de datos);*

*if(!$seleccion_bd)*

*echo "Error al seleccionar la BD.";*

## Funciones estado de la conexión a BD

Estas funciones permiten acceder a los valores de una conexión a una base de datos:

Para obtener información sobre la conexión usaremos:
*$info_con = mysqli_get_host_info($conexión_bd);*

Mientras que para recuperar las características del servidor MySQL tenemos:
*$info_serv = mysqli_get_server_info($conexión_bd);*

Las funciones siguientes nos ayudarán en la gestión de errores producidos en la conexión con la base de datos:

La primera nos dará número de error producido o 0 si no lo hay.

*$num_error = mysqli_connect_errno();*
La siguiente función nos devuelve una cadena con el error producido o cadena vacía.
*$msg_error = mysqli_connect_error();*

Ejemplo de uso de las funciones:

```
<?php
/* Ejemplo de conexión a una base de datos MySQL
 * Modificando los parámetros de conexión podemos comprobar
 * la sobrecarga de la función de conexión, quitando el
 * BASE_DATOS y luego usando la función mysqli_select_db
 */

  DEFINE ("SERVIDOR", "localhost");
  DEFINE ("USER", "root");
  DEFINE ("PASSWD", "root");
  DEFINE ("BASE_DATOS", "projectes");
```

```
  echo "Conexión DB:<br>";

  $conexion_bd = @mysqli_connect(SERVIDOR, USER, PASSWD, BASE_DATOS);
  if ($conexion_bd) {
      echo "Conexion correcta <br>";
      $info_con = mysqli_get_host_info($conexion_bd);
      echo "Información de conexión: " . $info_con . "<br>";
      $info_serv = mysqli_get_server_info($conexion_bd);
      echo "Información del servidor: " . $info_serv . "<br>";
      //Cierre de la conexión a la BD
      $cierre_bd = @mysqli_close($conexion_bd);
      if($cierre_bd){
            echo "Desconexión de la BD correcta <br>";
      }
      else {
            echo "Error al cerrar la conexión a la BD <br>";
      }
  }
  else {

      echo "Error de connexion con la BD<br>";
      $num_error = mysqli_connect_errno();
      echo "Número de error: " . $num_error . "<br>";
      $msg_error = mysqli_connect_error();
      echo "Mensaje error de conexión: " . $msg_error . "<br>";
   }
?>
```

# Acceso a datos

La operaciones relativas al acceso a datos las ejecutaremos mediante la función:

*$res = mysqli_query($conexión_db, sentencia sql);*
el resultado devuelto es, para una consulta un objeto de tipo *mysqli_result* o *false* si se produce un error y si la operación realizada es, por ejemplo un *insert*, el valor será booleano indicando el estado de finalización del proceso.

En caso de error podemos usar las funciones *mysqli_errno* y *mysqli_error* para documentar la situación producida.

## Consultas

Para conocer el número de líneas recuperadas en la consulta:

*$num_col = mysqli_num_rows($res);*

Las filas devueltas en la consulta se puede obtener usando varias funciones:

*$res_array = mysqli_fetch_array($res);*
esta función devuelve una línea de resultado y avanza el descriptor a la siguiente, el valor devuelto es un array en el que cada elemento corresponde a una columna de la fila actual. El índice del array puede ser tanto un entero o el nombre del campo de la tabla de la base de datos. Por ejemplo:
*echo "El identificador es: ". $res_array[0];* o bien: *echo "El identificador es: ". $res_array[id];*

*$res_array = mysqli_fetch_assoc($res);*
como la función anterior, devuelve un array con la línea actual y avanza el descriptor. El array es asociativo y el índice del array es el nombre de la columna.
*echo "El identificador es: ". $res_array[id];*

*$res_array = mysqli_fetch_row($res);*
esta función es como las dos anteriores pero el índice de acceso al array es numérico. Por ejemplo:
*echo "El identificador es: ". $res_array[0];*

*$res_obj = mysqli_fetch_object($res);*

con esta función la línea es devuelta en un objeto en el que cada atributo es una columna y su nombre corresponde al de la columna de la tabla. Como las anteriores, cada vez que se ejecuta avanza el descriptor a la línea siguiente del resultado. Por ejemplo:

*echo "El identificador es: ". $res_obj->id;*

Las funciones anteriores devuelven el valor *NULL* cuando ya no quedan más líneas en el resultado de la consulta.

*$res_array = mysqli_fetch_all($res[,MYSQLI_ASSOC o MYSQLI_NUM o MYSQLI_BOTH]);*

Esta función devuelve todas las líneas en un array de dos dimensiones, el primer índice es entero y corresponde al número de línea, mientras que el segundo podrá ser entero (*MYSQLI_NUM*) que es el valor por defecto en esta función, asociativo (*MYSQLI_ASSOC*) donde la segunda dimensión se accede mediante el nombre de la columna y finalmente con *MYSQLI_BOTH* se puede acceder mediante los dos valores entero o nombre de columna como en *mysqli_fetch_array*

```
<?php
  include "funcion_conexion_bd.php";

  DEFINE ("SERVIDOR", "localhost");
  DEFINE ("USER", "root");
  DEFINE ("PASSWD", "root");
  DEFINE ("BASE_DATOS", "projectes");

  $con_bd = conexion_bd(SERVIDOR, USER, PASSWD);
  // Con esta no usamos mysqli_select_db
  // $con_bd = conexion_bd(SERVIDOR, USER, PASSWD, BASE_DATOS);

  $cambio_ok = mysqli_select_db($con_bd, BASE_DATOS); //Selecciona la BD
  if($cambio_ok === TRUE){
    $sql = "SELECT * FROM projecte";
    if($res = mysqli_query($con_bd, $sql)) {
      $num_fil = mysqli_num_rows($res);
      echo "Núm. filas resultado: " . $num_fil ."<br>";
      $res_array = mysqli_fetch_array($res); // Acceso fila a fila
      while($res_array){
           print_r($res_array);
     echo "<br><br>";
     $res_array = mysqli_fetch_array($res);
      }
      mysqli_free_result($res);
    }
    else{
      echo "Error en la consulta: " . mysqli_error($con_bd) . "<br>";
    }
```

```
    //Todas la filas y acceso a los campos como array asociativo
    if($res = mysqli_query($con_bd, $sql)) {
      $res_array = mysqli_fetch_all($res, MYSQLI_ASSOC);

      for ($i=0; $i < $num_fil; $i++) {
      foreach($res_array[$i] as $clave => $valor){
          echo $clave . "-->" . $valor . "<br>";
        }
        echo "<br>";
      }
      mysqli_free_result($res);
    }
    else{
      echo "Error en la consulta: " . mysqli_error($con_bd) . "<br>";
    }
  }
  else{
      echo "Error en el cambio de BD: " . mysqli_error($con_bd) . "<br>";
  }
  mysqli_close($con_bd);
?>
```

Con la siguiente función podemos desplazar el descriptor de acceso al resultado de la consulta a la fila que no interese, por ejemplo, si se ha llegado al final para volver al principio (desplazamiento 0) y reutilizar los datos. El valor devuelto es booleano en función de si se ha ejecutado correctamente.

*$desplazamiento_error =* ***mysqli_data_seek*** *( $res , $desplazamiento );*

por ejemplo en el código anterior, no sería necesario ejecutar la consulta de nuevo con:

```
      mysqli_data_seek($res, 0); // Pone el descriptor al principio
      $res_array = mysqli_fetch_all($res, MYSQLI_ASSOC);

      for ($i=0; $i < $num_fil; $i++) {
         foreach($res_array[$i] as $clave => $valor){
           echo $clave . "-->" . $valor . "<br>";
         }
         echo "<br>";
      }
      mysqli_free_result($res);
```

En caso de ejecutar operaciones que modifiquen el contenido de la base de datos, como *insert*, *update* o *delete*, podemos usar la función *mysqli_affected_rows* para comprobar el número de filas accedidas o -1 en caso de error.

$num_filas = **mysqli_affected_rows** ( $con_bd );

## Liberar el resultado de una consulta

Cuando ya hemos accedido a la información de una consulta es conveniente liberar el espacio que ocupa con:

*mysqli_free_result($res);*
esta función no devuelve ningún resultado y el parámetro es un objeto devuelto por *mysqli_query.*

## Información columnas de la consulta

Podemos conocer el número de columnas presentes en el resultado de la consulta actual mediante la función:

$num_colum = **mysqli_num_fields** ( $res );

La siguiente función nos indica el número de la columna en la que se encuentra el descriptor de acceso al resultado de la consulta que estamos tratando:

$colum = **mysqli_field_tell** ( $res );

Y podemos posicionar el descriptor en la columna que nos interese (para acceder a la primera columa 0) con :

$pos_column_error = **mysqli_field_seek** ($res , $colum );

Con la siguiente función obtenemos un objeto con toda la información de la estructura de la columna actual, tal como su nombre, tipo de dato, longitud máxima, tabla, base de datos, etc.

$obj_colum = **mysqli_fetch_field** ( $res );

Los códigos numéricos devueltos en los atributos se pueden consultar en la web de PHP **mysqli_fetch_field** por ejemplo 253 es VARCHAR o 13 YEAR.

Modificando el ejemplo anterior, podemos ver cómo usar la funciones anteriores:

```
  $sql = "SELECT * FROM projecte";
    if($res = mysqli_query($con_bd, $sql)) {
      // Acceso a las columnas del resultado
      $num_colum = mysqli_num_fields($res);
      echo "Número de campos: " . $num_colum . "<br>";
      for($i = 0; $i < $num_colum ; $i++){
        $obj_colum = mysqli_fetch_field($res);
           echo "Columna núm: " . mysqli_field_tell($res) . " Tabla: " .
$obj_colum->table . " nombre campo: " . $obj_colum->name . " tipo: " .
$obj_colum->type . "<br>";

      }
      mysqli_free_result($res);
    }
    else{
      echo "Error en la consulta: " . mysqli_error($con_bd) . "<br>";
    }
```

# PDO – Objetos de datos PHP

PDO es la extensión de objetos de datos en PHP que nos facilita la comunicación con bases de datos. Hay diferentes implementaciones para acceder a las correspondientes SGBD. De forma que se usa el controlador específico PDO para el SGBD que tiene las bases de datos a las que vamos a acceder.

La ventaja de PDO es que las funciones para operar con las bases de datos son siempre las mismas independientemente de si el SGBD es MySQL, PostgreSQL, Oracle, etc.

PDO es una clase que instanciamos para obtener la conexión de acceso a la base de datos, debemos indicar, además de los parámetros habituales: usuario, contraseña y base de datos, el tipo de SGBD que vamos a usar:

```
try{
   $con = new PDO('mysql:host=IP o nom_server; dbname= nom_bd', $user,$passwd);
   ....
}catch (PDOException $e) {
   echo "Error: " . $e→getMessage() . "<br>";
}
```

Si se produce algún error al establecer la conexión, se genera una *PDOException* que deberá ser tratada.

En caso que la conexión se establezca correctamente, se devuelve un objeto PDO y la conexión estará disponible mientras este objeto mantenga su valor almacenado en variables o referencias. La conexión la cerraremos eliminado el objeto PDO, por ejemplo asignando *null* a su variable y si hay más referencias a este objeto también deberán ser eliminadas. Esta conexión también se cierra al finalizar la ejecución.

PDO también permite la persistencia en las conexiones a bases de datos, que no se cierren al finalizar la ejecución y puedan ser reutilizadas en nuevas ejecuciones de la aplicación. Estas conexiones se guardan en una caché y se reutilizan cuando se ejecuta una nueva conexión con los mismos parámetros, evitando así la sobrecarga que supone abrir y cerrar las conexiones a las bases de datos. Para activar la persistencia la apertura de la conexión quedará como:

```
try{
  $con = new PDO('mysql:host=IP o nom_server; dbname= nom_bd', $user,$passwd,
array(PDO::ATTR_PERSISTENT => true));
  ....
}catch (PDOException $e) {
  echo "Error: " . $e→getMessage() . "<br>";
}
```

La clase PDO dispone de los métodos para operar con la base de datos:

`$con→query(string $sql);` ejecuta la consulta (select) especificada en $sql, el valor devuelto será un objeto de la clase *PDOStatement* para acceder a los resultados de la consulta o *false* se se produce un error. Si se necesita una consulta que se ejecute repetidas veces, se recomienda usar *PDO::prepare* junto con *PDOStatement::execute*.

Para realizar operaciones de inserción, actualización y borrado se usa el método *PDO::exec* :

`$num_filas = $con→exec($sql);` que devuelve el número de filas afectadas en la operación. O *false* si se produce un error, por tanto se recomienda evaluarla con `===` ya que la ejecución normal puede devolver 0 filas y confundirse con false.

Los *resultset* devueltos en las consultas hemos dicho que se guardan en objetos PDOStatement y con sus métodos podemos acceder a los datos, destacaremos:

*PDO::closeCursor():bool* que cierra el cursor y permite que la sentencia sea ejecutada nuevamente.

*PDO::columnCount():int* da el número de columnas del *resultset*.

*PDO::fetch()* devuelve la siguiente fila del *resultset*, en caso de error *false*. A este método se le puede pasar un parámetro para indicar cómo se devuelve la fila de datos:

- PDO::FETCH_ASSOC devuelve un array asociativo donde el índice es el nombre de las columnas.
- PDO::FETCH_OBJ devuelve un objeto donde los nombres de las columnas corresponde a sus atributos.
- PDO::FETCH_NUM devuelve un array con índice numérico.
- PDO::FETCH_BOTH (valor por defecto) devuelve un array indexado tanto de forma asociativa como numérica.

*PDO::fetch_all()* devuelve un array bidimensional con todas las filas del *resulset*, como en *fetch*() podemos pasarle uno de los parámetros anteriores para indicar cómo queremos acceder a los datos.

*PDO::rowCount():int* devuelve el número de filas obetenidas.

Además, PDO dispone de una clase para la gestión de errores y excepciones, se trata de *PDOException*. Al usar PDO debemos poner sus instrucciones en cláusulas *try-catch* para que se puedan gestionar sus errores. En esta clase usaremos los métodos getMessage() y getCode() para obtener los mensajes y códigos de error.

Ejemplo de acceso con PDO a una base de datos:

```
<?php

/*
 * Juego de lanzar una moneda y elegir cara o cruz, el programa simula la moneda
 * con la función random()%2 0--> cara y 1--> cruz
 * El programa guarda en la BD caracruz el récord de partidas ganadas de forma
 * seguida, si se supera el record actual se almacena el nuevo record en la BD
 * Para conservar el número de partidas ganadas entre llamadas usaremos una
 * una variable hidden.
 */

DEFINE ("SERVIDOR", "localhost");
DEFINE ("USER", "aboronat");
DEFINE ("PASSWD", "aboronat");
DEFINE ("BASE_DATOS", "caracruz");


if (!isset($_POST["record"])){
    $_POST["record"] = 0;
    $_POST["BD"] = false;
    $_POST["record_actual"] = 0;
}
if ($_POST["BD"] === false){
  try{
    $con_bd = new PDO('mysql:host=' . SERVIDOR . ';dbname=' . BASE_DATOS , USER,
PASSWD);
    $sql = "SELECT * FROM record_cara_cruz WHERE id = 1" ;
    $res = $con_bd->query($sql);
    if($res !== false){
       $res_array = $res->fetch(PDO::FETCH_ASSOC);
       $_POST["record_actual"] = $res_array["record"];
       $_POST["BD"] = true;
    }
  }catch (PDOException $e){
      echo "Error acceso a la bd: " . $e->getMessage() . "<br>";
  }
}
$resultado = "";
$aux = random_int(0, 9) % 2;
```

```
switch ($aux){
    case 0 : $moneda = " CARA ";
             BREAK;
    case 1 : $moneda = " CRUZ ";
             BREAK;
}
if(isset($_POST["cara"])){
    if($aux == 0){
        $resultado = " GANAS<br>";
        $_POST["record"]++;
    }
    else {
        $resultado = " PIERDES<br>";
        $_POST["record"] = 0;
    }
}
if(isset($_POST["cruz"])){
    if($aux == 1){
        $resultado = " GANAS<br>";
        $_POST["record"]++;
    }
    else {
        $resultado = " PIERDES<br>";
        $_POST["record"] = 0;
    }
}
if($_POST["record"] > $_POST["record_actual"]){
  try{
    $_POST["record_actual"] = $_POST["record"];
    $con_bd = new PDO('mysql:host=' . SERVIDOR . ';dbname=' . BASE_DATOS, USER,
PASSWD);
    $sql = "UPDATE record_cara_cruz SET record = " . $_POST["record"] . " WHERE
id = 1 ";
    $res = $con_bd->exec($sql);
    if($res > 0) {
       echo "Record registrado <br>";
    }
    else{
       echo "Error en la consulta<br>";
    }
  }catch(PDOException $e){
      echo "Error acceso a la bd: " . $e->getMessage() . "<br>";
  }
}
?>

<!DOCTYPE html>
<!--
 Formulario para jugar a cara o cruz
-->
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
    </head>
```

```
<body>
<form name="autenticar" method="post" action="<?php echo $_SERVER['PHP_SELF']; ?
>" >
    <?php echo "Serie de partidas ganadas: " .$_POST["record"] . "  Record
actual: " . $_POST["record_actual"] . "<br>"; ?>
    <?php echo "Ha salido:  " . $moneda . "  tu  " . $resultado . "<br>"; ?>
    <input type="hidden" name="record" value="<?php echo $_POST['record']; ?>">
    <input type="hidden" name="BD" value="<?php echo $_POST['BD']; ?>">
    <input type="hidden" name="record_actual" value="<?php echo
$_POST['record_actual']; ?>">
    <input type="submit" name="cara" value="Cara">
    <input type="submit" name="cruz" value="Cruz">
</form>
```