Conexión con MySQL

En el caso de MySQL, para crear el objeto PDO se necesita proporcionar el nombre del servidor, el nombre de usuario y la contraseña. En el ejemplo siguiente esos datos se proporcionan como constantes que deberían definirse en el programa.

```
// FUNCIÓN DE CONEXIÓN CON LA BASE DE DATOS MYSQL
function conectaDb()
{
    try {
        $tmp = new PDO(MYSQL_HOST, MYSQL_USER, MYSQL_PASSWORD);
        $tmp->setAttribute(PDO::MYSQL_ATTR_USE_BUFFERED_QUERY, true);
        $tmp->exec("set names utf8mb4");
        return($tmp);
    } catch(PDOException $e) {
        cabecera("Error grave", MENU_PRINCIPAL);
        print "<p>Error: No puede conectarse con la base de
datos.</p>\n";
        print "\n";
        print "    <p>Error: " . $e->getMessage() . "</p>\n";
        pie();
        exit();
    }
}

// EJEMPLO DE USO DE LA FUNCIÓN conectaDB()
// La conexión se debe realizar en cada página que acceda a la base de
datos
$db = conectaDB();
```

Desconexión con la base de datos

Para desconectar con la base de datos hay que destruir el objeto PDO. Si no se destruye el objeto PDO, PHP lo destruye al terminar la página.

```
$db = null;
```

Consultas a la base de datos

Una vez realizada la conexión a la base de datos, las operaciones se realizan a través de consultas.

El método para efectuar consultas es **PDO->query($consulta)**, que devuelve el resultado de la consulta. Dependiendo del tipo de consulta, el dato devuelto debe tratarse de formas distintas.

- Si es una consulta que no devuelve registros, sino que simplemente realiza una acción que puede tener éxito o no (por ejemplo, insertar un registro), el método devuelve **true** o **false**. No es necesario guardar el resultado de la consulta en ninguna variable, pero se puede utilizar para sacar un mensaje diciendo que todo ha ido bien (o no).

```
// EJEMPLO DE CONSULTA DE INSERCIÓN DE REGISTRO
require_once "biblioteca.php";
$db = conectaDB();

$consulta = "INSERT INTO $dbTabla
    (nombre, apellidos)
    VALUES ("$nombre", "$apellidos")";
if ($db->query($consulta)) {
    print "    <p>Registro creado correctamente.</p>\n";
} else {
    print "    <p>Error al crear el registro.<p>\n";
}

$db = null;
```

Pero si la consulta devuelve registros, el método devuelve los registros correspondientes o **false**. En ese caso sí que es conveniente guardar lo que devuelve el método en una variable para procesarla posteriormente. Si contiene registros, la variable es de un tipo especial llamado recurso que no se puede acceder directamente, pero que se puede recorrer con un bucle **foreach()**,

```
// EJEMPLO DE CONSULTA DE SELECCIÓN DE REGISTROS
require_once "biblioteca.php";
$db = conectaDB();

$consulta = "SELECT * FROM $dbTabla";
$result = $db->query($consulta);
if (!$result) {
    print "    <p>Error en la consulta.</p>\n";
} else {
    foreach ($result as $valor) {
        print "    <p>$valor[nombre] $valor[apellidos]</p>\n";
    }
}

$db = null;
```

Seguridad en las consultas: consultas preparadas

Para evitar ataques de inyección SQL se recomienda el uso de sentencias preparadas, en las que PHP se encarga de "desinfectar" los datos en caso necesario. En general, cualquier consulta que incluya datos introducidos por el usuario debe realizarse mediante consultas preparadas.

Consultas preparadas

El método para efectuar consultas es primero preparar la consulta con **PDO->prepare($consulta)** y después ejecutarla con **PDO->execute([parámetros])**, que devuelve el resultado de la consulta.

```
// Consulta preparada
$consulta = "SELECT * FROM $dbTabla";
$result = $db->prepare($consulta);
$result->execute();
```

Dependiendo del tipo de consulta, el dato devuelto debe tratarse de formas distintas, como se ha explicado en el apartado anterior.

Si la consulta incluye datos introducidos por el usuario, los datos pueden incluirse directamente en la consulta, pero en ese caso, PHP no realiza ninguna "desinfección" de los datos, por lo que estaríamos corriendo riesgos de ataques:

```
$nombre    = $_REQUEST["nombre"];
$apellidos = $_REQUEST["apellidos"];

$consulta = "SELECT COUNT(*) FROM $dbTabla
    WHERE nombre=$nombre
    AND apellidos=$apellidos"; // DESACONSEJADO: PHP NO DESINFECTA
LOS DATOS
$result = $db->prepare($consulta);
$result->execute();
if (!$result) {
    print "    <p>Error en la consulta.</p>\n";
 ...
```

Para que PHP desinfecte los datos, estos deben enviarse al ejecutar la consulta, no al prepararla. Para ello es necesario indicar en la consulta la posición de los datos. Esto se puede hacer de dos maneras, mediante parámetros o mediante interrogantes, aunque se aconseja la utilización de parámetros:

- mediante parámetros (:parametro)

  En este caso la matriz debe incluir los nombres de los parámetros y los valores que sustituyen a los parámetros (el orden no es importante), como muestra el siguiente ejemplo:

  ```
  $nombre    = $_REQUEST["nombre"];
  $apellidos = $_REQUEST["apellidos"];

  $consulta = "SELECT COUNT(*) FROM $dbTabla
      WHERE nombre=:nombre
  ```

```
    AND apellidos=:apellidos";
$result = $db->prepare($consulta);
$result->execute([":nombre" => $nombre, ":apellidos" =>
$apellidos]);
if (!$result) {
    print "    <p>Error en la consulta.</p>\n";
 ...
```

- mediantes interrogantes (?)

  En este caso la matriz debe incluir los valores que sustituyen a los interrogantes en el mismo orden en que aparecen en la consulta, como muestra el siguiente ejemplo:

```
$nombre    = $_REQUEST["nombre"];
$apellidos = $_REQUEST["apellidos"];

$consulta = "SELECT COUNT(*) FROM $dbTabla
    WHERE nombre=?
    AND apellidos=?";
$result = $db->prepare($consulta);
$result->execute([$nombre, $apellidos]);
if (!$result) {
    print "    <p>Error en la consulta.</p>\n";
 ...
```

Solución configurable

El resultado sería

en el fichero que quiera reiniciar la base de datos se llamaría a la biblioteca general y se llamaría a la función genérica borraTodo():

```
// EJEMPLO DE USO DE CONEXIÓN CONFIGURABLE
// La conexión se debe realizar en cada página que acceda a la base de datos
require_once "biblioteca.php";
$db = conectaDb();
borraTodo($db);
$db = null;
```

biblioteca-mysql.php: contiene la definición de la función borraTodo() específica para trabajar con MySQL y que borra la base de datos, la crea y crea la tabla:

```
// biblioteca-mysql.php
$consultaCreaTabla = "CREATE TABLE $dbTabla (
    id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
    nombre VARCHAR($tamNombre),
    apellidos VARCHAR($tamApellidos),
    PRIMARY KEY(id)
    )";

function borraTodo($db)
{
    global $dbDb, $consultaCreaTabla;

    $consulta = "DROP DATABASE $dbDb";
    if ($db->query($consulta)) {
        print "    <p>Base de datos borrada correctamente.</p>\n";
        print "\n";
    } else {
        print "    <p>Error al borrar la base de datos.</p>\n";
        print "\n";
    }
    $consulta = "CREATE DATABASE $dbDb";
    if ($db->query($consulta)) {
        print "    <p>Base de datos creada correctamente.</p>\n";
        print "\n";
        $consulta = $consultaCreaTabla;
        if ($db->query($consulta)) {
            print "    <p>Tabla creada correctamente.</p>\n";
            print "\n";
        } else {
            print "    <p>Error al crear la tabla.</p>\n";
            print "\n";
        }
    } else {
        print "    <p>Error al crear la base de datos.</p>\n";
```

```
        print "\n";
    }
      }
```

Consultas DROP TABLE, INSERT INTO, UPDATE, DELETE FROM

Para borrar una tabla, se utiliza la consulta DROP TABLE.

```
// EJEMPLO DE CONSULTA DE BORRADO DE TABLA
$consulta = "DROP TABLE $dbTabla";
if ($db->query($consulta)) {
    print "    <p>Tabla borrada correctamente.</p>\n";
    print "\n";
} else {
    print "    <p>Error al borrar la tabla.</p>\n";
    print "\n";
}
```

Para añadir un registro a una tabla, se utiliza la consulta INSERT INTO.

```
// EJEMPLO DE CONSULTA DE INSERCIÓN DE REGISTRO
$nombre    = recoge("nombre");
$apellidos = recoge("apellidos");

$consulta = "INSERT INTO $dbTabla
    (nombre, apellidos)
    VALUES (:nombre, :apellidos)";
$result = $db->prepare($consulta);
if ($result->execute([":nombre" => $nombre, ":apellidos" =>
$apellidos])) {
    print "    <p>Registro creado correctamente.</p>\n";
    print "\n";
} else {
    print "    <p>Error al crear el registro.</p>\n";
    print "\n";
}
```

Para modificar un registro a una tabla, se utiliza la consulta UPDATE.

```
// EJEMPLO DE CONSULTA DE MODIFICACIÓN DE REGISTRO
$nombre    = recoge("nombre");
$apellidos = recoge("apellidos");
$id        = recoge("id");

$consulta = "UPDATE $dbTabla
    SET nombre=:nombre, apellidos=:apellidos
    WHERE id=:id";
$result = $db->prepare($consulta);
if ($result->execute([":nombre" => $nombre, ":apellidos" =>
$apellidos, ":id" => $id])) {
    print "    <p>Registro modificado correctamente.</p>\n";
    print "\n";
} else {
    print "    <p>Error al modificar el registro.</p>\n";
    print "\n";
}
```

Para borrar un registro de una tabla, se utiliza la consulta DELETE FROM.

Nota: En el ejemplo, los registros a borrar se reciben en forma de matriz y se recorre la matriz borrando un elemento en cada iteración.

```
// EJEMPLO DE CONSULTA DE BORRADO DE REGISTRO
$id = recogeMatriz("id");

foreach ($id as $indice => $valor) {
    $consulta = "DELETE FROM $dbTabla
        WHERE id=:indice";
    $result = $db->prepare($consulta);
    if ($result->execute([":indice" => $indice])) {
        print "    <p>Registro borrado correctamente.</p>\n";
        print "\n";
    } else {
        print "    <p>Error al borrar el registro.</p>\n";
        print "\n";
    }
}
```

Consulta SELECT

Para obtener registros que cumplan determinados criterios se utiliza una consulta SELECT.

Si se produce un error en la consulta, la consulta devuelve el valor **false** .

```
    // EJEMPLO DE CONSULTA DE SELECCIÓN DE REGISTROS
$consulta = "SELECT * FROM $dbTabla";
$result = $db->query($consulta);
if (!$result) {
    print "    <p>Error en la consulta.</p>\n";
    print "\n";
} else {
    print "    <p>Consulta ejecutada.</p>\n";
    print "\n";
}
```

Si la consulta devuelve un único registro se puede utilizar la función [PDOStatement->fetchColumn()] para recuperar la primera columna.

```
// EJEMPLO DE CONSULTA DE SELECCIÓN DE REGISTROS
$consulta = "SELECT COUNT(*) FROM $dbTabla";
$result = $db->query($consulta);
if (!$result) {
    print "    <p>Error en la consulta.</p>\n";
    print "\n";
} else {
    $encontrados = $result->fetchColumn();
    print "    <p>Se han encontrado $encontrados registros.</p>\n";
```

```
    print "\n";
}
```

Si la consulta se ejecuta correctamente, la consulta devuelve los registros correspondientes.

Para acceder a los registros devueltos por la consulta, se puede utilizar un bucle foreach.

```
// EJEMPLO DE CONSULTA DE SELECCIÓN DE REGISTROS
$consulta = "SELECT * FROM $dbTabla";
$result = $db->query($consulta);
if (!$result) {
    print "    <p>Error en la consulta.</p>\n";
    print "\n";
} else {
    foreach ($result as $valor) {
        print "    <p>Nombre: $valor[nombre] - Apellidos:
$valor[apellidos]</p>\n";
        print "\n";
    }
}
```

O también se puede utilizar la función **PDOStatement->fetch()**.

```
// EJEMPLO DE CONSULTA DE SELECCIÓN DE REGISTROS
$consulta = "SELECT * FROM $dbTabla";
$result = $db->query($consulta);
if (!$result) {
    print "    <p>Error en la consulta.</p>\n";
    print "\n";
} else {
    while ($valor = $result->fetch()) {
        print "    <p>Nombre: $valor[nombre] - Apellidos:
$valor[apellidos]</p>\n";
        print "\n";
    }
}
```

- Si la consulta no devuelve ningún registro, los dos bucles anteriores (foreach o fetch) no escribirían nada. Por ello se recomienda hacer primero una consulta que cuente el número de resultados de la consulta y, si es mayor que cero, hacer la consulta.

  El ejemplo siguiente utiliza la función **PDOStatement->fetchColumn()**, que devuelve la primera columna del primer resultado (que en este caso contiene el número de registros de la consulta).

```
// EJEMPLO DE CONSULTA DE SELECCIÓN DE REGISTROS
$consulta = "SELECT COUNT(*) FROM $dbTabla";
$result = $db->query($consulta);
if (!$result) {
    print "<p>Error en la consulta.</p>\n";
    print "\n";
} elseif ($result->fetchColumn() == 0) {
    print "<p>No se ha creado todavía ningún registro en la
tabla.</p>\n";
```

```
        print "\n";
    } else {
        $consulta = "SELECT * FROM $dbTabla";
        $result = $db->query($consulta);
        if (!$result) {
            print "    <p>Error en la consulta.</p>\n";
            print "\n";
        } else {
            foreach ($result as $valor) {
                print "    <p>Nombre: $valor[nombre] - Apellidos:
    $valor[apellidos]</p>\n";
                print "\n";
            }
        }
    }
```

Consulta SELECT LIKE

La consulta SELECT permite efectuar búsquedas en cadenas utilizando el condicional LIKE o NOT LIKE y los comodines _ (cualquier carácter) o % (cualquier número de caracteres).

Ejemplos de consultas:

Registros en los que el apellido empieza por la cadena recibida:

```
// EJEMPLO DE CONSULTA DE SELECCIÓN DE REGISTROS
$apellidos = recoge("apellidos");

$consulta = "SELECT COUNT(*) FROM $dbTabla
    WHERE apellidos LIKE :apellidos";
$result = $db->prepare($consulta);
$result->execute([":apellidos" => "$apellidos%"]);
if (!$result) {
    print "    <p>Error en la consulta.</p>\n";
    print "\n";
} else {
    print "    <p>Se han encontrado " . $result->fetchColumn() . "
registros.</p>\n";
    print "\n";
}
```