

# UD-04b: Acceso a BD con PHP

Desarrollo Web en Entorno Servidor

Curso 2020/2021

# Utilización de MySQL en PHP

## Consultas preparadas

- Sirven para evitar tener que repetir aquellas sentencias SQL que se repiten de forma habitual en un programa, como las que insertan valores en una tabla.
- Estas consultas se almacenan en el servidor para ser ejecutadas cuando sea necesario.
- Además tienen ciertas ventajas, como evitar inyecciones de código SQL, o que con consultas del tipo INSERT son más rápidas y eficientes.

Las consultas preparadas usan parámetros para preparar una consulta: en lugar de poner los valores se indica, con **signos de interrogación**, su posición dentro de la sentencia.

```
// $conexion->prepare devuelve un objeto de tipo mysqli_stmt
$resultado = $conexion->prepare("INSERT INTO Clientes VALUES (?, ?, ?)");

// Asigna entero, string, entero a la consulta anterior
$resultado->bind_param('isi', $id, $nombre, $telefono);
```

# Utilización de MySQL en PHP

## Ejemplo

Una BD llamada *bdproductos* y una tabla llamada *productos* con estos campos...

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	<b>codigo</b> 🏠	int(11)			No	Ninguna		AUTO_INCREMENT
2	<b>seccion</b>	varchar(50)	utf8_general_ci		No	Ninguna		
3	<b>articulo</b>	varchar(100)	utf8_general_ci		No	Ninguna		
4	<b>precio</b>	int(5)			No	Ninguna		
5	<b>paisorigen</b>	varchar(50)	utf8_general_ci		No	Ninguna		

Busca un checkbox llamado 'A\_I'

### formulariobusqueda.php

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>Países</title>
</head>
<body>
    <form action="resultadopaises.php" method="get">
        <label>Escribe el país: <input type="text" name="pais" /></label>
        <input type="submit" name="enviando" value="BUSCAR" />
    </form>
</body>
</html>
```

Tenemos 3 archivos:

### datosconexion.php

```
<?php
$db_host="localhost";
$db_usuario="root";
$db_contrasena="";
$db_nombre="bdproductos";
?>
```

## resultadopaises.php

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>Paises</title> </head>
<body>
<?php
    require ("datosconexion.php");

    $conexion = mysqli_connect($db_host, $db_usuario, $db_contrasena);

    if (mysqli_connect_errno()){
        echo "Houston, tenemos un problema...";
        exit();
    }
    // seleccionamos la conexión y la base de datos (si verdadero, ya no muere)
    mysqli_select_db($conexion, $db_nombre) or die ("NO encuentro la base de datos.");

    // para admitir ñ y tildes (conjunto de caracteres para envíos desde y hacia el servidor de la BD).
    mysqli_set_charset($conexion, "utf8");

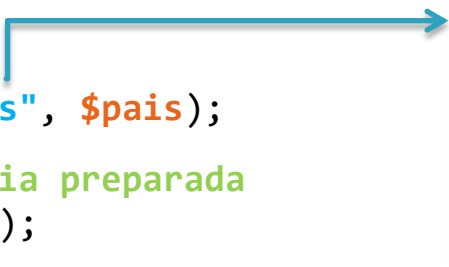
    // creamos la sentencia SQL sustituyendo los criterios por ?
    $sql = "SELECT codigo, seccion, precio, paisorigen FROM productos WHERE paisorigen = ? ";

    $resultado = mysqli_prepare($conexion, $sql);

    $pais = $_GET["pais"];

    mysqli_stmt_bind_param($resultado, "s", $pais);

    // finalmente, ejecutamos la sentencia preparada
    $ok = mysqli_stmt_execute($resultado);
```



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 y se envía en paquetes



```
if (!$ok){
    echo "Error al ejecutar la consulta";
}else{

    // asociamos variables nuevas al resultado de la consulta
    mysqli_stmt_bind_result($resultado, $micodigo, $miseccion, $miprecio, $mipais);

    // mostramos por pantalla los valores encontrados
    echo "Artículos Encontrados: <br><br>";

    // recorremos las filas del resultado
    while (mysqli_stmt_fetch($resultado))
    {
        echo $micodigo ." ". $miseccion ." ". $miprecio ." ". $mipais."<br>";
    }

    //IMPORTANTE: Cerramos el objeto $resultado
    mysqli_stmt_close($resultado);
}
// cerramos conexión
mysqli_close($conexion);
```

?>

```
</body>
</html>
```



## EJERCICIO

Usa el ejemplo de **productos** anterior y crea una nueva página para insertar nuevos productos mediante consulta preparada.

Se deben **todos** los ficheros dentro de una carpeta llamada **ud04ejer03**:

- ✓ formulariointroduccion.php
- ✓ resultadoinsertar.php
- ✓ datosconexion.php
- ✓ formulariobusqueda.php
- ✓ resultadopais.php

Combina los archivos de este ejercicio para poder ir navegando de una opción a otra.

formulariointroduccion.php

### Dar de alta ARTÍCULOS NUEVOS

Sección del Artículo   
Nombre del Artículo   
Precio del Artículo   
País Origen Artículo

INSERTAR!!

Esta vez, se pueden dejar campos vacíos

resultadoinsertar.php

Artículo Insertado....

[Insertar un Artículo NUEVO](#)

Aquí debes mostrar:

Artículo \$nombre insertado en \$seccion.

# Utilización de PDO con MySQL en PHP

## PHP Data Objects (PDO)

- Si vamos a programar una aplicación usando MySQL como SGBD, la extensión MySQLi es una buena opción. Ofrece acceso a todas las características del motor de BD, a la vez que reduce los tiempos de espera en la ejecución de sentencias.
- Sin embargo, si en el futuro tuviéramos que cambiar de SGBD, tendríamos que volver a programar el código. Siendo así, debemos adoptar una capa de abstracción para el acceso a los datos. Para ello, una opción muy recomendable es PDO.
- Aunque la principal característica de PDO es que se puede usar con la mayoría de SGDBs, ciertos detalles han de tenerse en cuenta. Por ejemplo, no modifica las sentencias SQL para adaptarlas a las características específicas de cada servidor.
- La extensión PDO debe usar un driver o controlador específico para el tipo de BD que se utilice. Para ver los drivers instalados consulta la información en *phpinfo()*.
- Para acceder a las funcionalidades de PDO tenemos que usar los objetos que ofrece, con sus métodos y propiedades. No existen funciones alternativas.

# Utilización de PDO con MySQL en PHP

## Ejecución de consultas que NO devuelven datos

Para consultas de acción, que no devuelven datos, como INSERT, DELETE o UPDATE, utilizaremos la función `exec`, que además devuelve el nº de registros afectados.

```
$rows = $conexion->exec('DELETE FROM stock WHERE unidades = 0');  
echo "<p>Se han borrado $rows registros.</p>";
```

**Recuerda:** Con mysqli era muy distinto. En `$rows` se guardaba *true/false* y después podíamos llamar a `$conexion->affected_rows`.



# Utilización de PDO con MySQL en PHP

## Ejecución de consultas que SÍ devuelven datos

Si la consulta genera un conjunto de datos, como es el caso de **SELECT**, usaremos el método `query`, como en *mysqli*. Devuelve un objeto de la clase *PDOStatement*.

```
$consulta = $conexion->query("SELECT producto, unidades FROM stock");
```

Para tratar el conjunto de resultados obtenido, podemos usar el método `fetch` de la clase *PDOStatement*, que devuelve un registro (fila) del conjunto de resultados o `false` si ya no quedan registros por recorrer.

```
while ($resultado = $consulta->fetch())  
{  
    // Aquí accederíamos a $resultado['producto'] y $resultado['unidades']  
}  
/* Este fetch() de PDO devuelve resultados como el fetch_array de  
   mysqli, es decir, tanto array numérico como asociativo. */
```

# Utilización de PDO con MySQL en PHP

## Parámetros del método *fetch*

Por defecto, el método *fetch* devuelve un array con claves numéricas y asociativas. Para cambiar su comportamiento, admite un parámetro opcional que puede tomar uno de los siguientes valores:

- **PDO::FETCH\_ASSOC** → Devuelve sólo un array asociativo.
- **PDO::FETCH\_NUM** → Array con claves numéricas.
- **PDO::FETCH\_BOTH** → Usado por defecto. Array con claves numéricas y asociativas.
- **PDO::FETCH\_OBJ** → Objeto cuyas propiedades se corresponden con los campos del registro.
- **PDO::FETCH\_LAZY** → Tanto el objeto como el array con clave dual anterior (todo).
- **PDO::FETCH\_BOUND** → Devuelve true y asigna los valores del registro a variables, según se indique con el método *bindColumn*. Este método debe ser llamado una vez por cada columna, indicando en cada llamada el número de columna (empezando en 1) y la variable a asignar.



# Utilización de PDO con MySQL en PHP

Antes de ejecutar la consulta, **asignamos valores** a los parámetros **con** `bindParam`. Dependerá de si habíamos usado interrogaciones (?) o nombres (:*nombre*)

Tal y como sucedía en MySQLi, al usar `bindParam` para asignar los parámetros de una consulta preparada, **se deben usar siempre variables**.

```
$cod_producto = "TABLET";  
$nombre_producto = "Tablet PC";
```

(?, ?) {  
    `$consulta->bindParam(1, $cod_producto);`  
    `$consulta->bindParam(2, $nombre_producto);`

(:cod, :nombre) {  
    `$consulta->bindParam(":cod", $cod_producto);`  
    `$consulta->bindParam(":nombre", $nombre_producto);`

**Recuerda:** en mysqli es `bind_param`, con barra baja. Se usa de forma distinta.

# Utilización de PDO con MySQL en PHP

Una vez preparada la consulta y enlazados los parámetros con sus valores, se ejecuta la consulta con el método `execute`.

```
$consulta->execute();
```

También es posible asignar los valores de los parámetros en el momento de ejecutar la consulta, con un array (asociativo o con claves numéricas dependiendo de la forma en que hayas indicado los parámetros) en la llamada a `execute`.


```
$arrayParametros = array(":cod" => "TABLET", ":nombre" => "Tablet PC");  
$consulta->execute($arrayParametros);
```

# Utilización de PDO con MySQL en PHP

**EJEMPLO:** Una BD llamada *productosbeni*, con una tabla llamada *productos*.

Realizaremos este ejemplo, como una **aproximación al patrón 'Modelo-Vista-Controlador'**.

- Crea 2 carpetas: **Controlador** y **Modelo**.
- En la carpeta Modelo, tendremos dos archivos: `clase_conexion.php` y `clase_consulta.php`.



```
<?php
class Conexion
{
    public function getConexion()
    {
        $host = "localhost";
        $db = "productosbeni";
        $user = "root";
        $pass = "";

        $conexion = new PDO("mysql:host=$host;dbname=$db;", $user, $pass);
        return $conexion;
    }
}
?>
```

clase\_consulta.php

Este método hace el INSERT y devuelve un mensaje indicando cómo ha ido.

<?php

```
class Consulta
{
    public function insertarProducto($minom,$midesc,$micat,$miprec)
    {
        $conexion= new Conexion();
        $pdoObject = $conexion->getConexion();


        $sql = "INSERT INTO productos (nombre, descripcion, categoria, precio)
                VALUES (:nombre, :descripcion, :categoria, :precio)";

        $sentencia = $pdoObject->prepare($sql);

        $sentencia->bindParam(':nombre', $minom);
        $sentencia->bindParam(':descripcion', $midesc);
        $sentencia->bindParam(':categoria', $micat);
        $sentencia->bindParam(':precio', $miprec);

        if (@$sentencia->execute()) {
            $mensaje = "Registro creado correctamente.";
        } else {
            $mensaje = "Fallo al crear el registro.";
        }
        return $mensaje;
    }
}
```





Este método hace la  
SELECT y devuelve el  
resultado en un array.

```
public function cargarProductos()  
{  
    $modelo = new Conexion();  
    $conexion = $modelo->getConexion();  
    $sql = "SELECT * FROM productos";  
    $resultado = $conexion->prepare($sql);  
    $resultado->execute();  
    $rows = null;  
    while ($fila = $resultado->fetch())  
    {  
        // guardamos las filas en un array  
        $rows[] = $fila;  
    }  
    return $rows;  
}
```

?>



# Utilización de PDO con MySQL en PHP

En la carpeta **Controlador**, tendremos 2 archivos: `cargar.php` e `insertar.php`.

Fuera de estas dos carpetas, ‘vemos’ otros 2 archivos: `insertar.html` y `verproductos.php`.

*Se quedaría algo así:*

```
▼ EJEMPLO-DE-PDO
  ▼ Controlador
    🐘 cargar.php
    🐘 insertar.php
  ▼ Modelo
    🐘 clase_conexion.php
    🐘 clase_consulta.php
  <> insertar.html
  📄 productosbeni.sql
  🐘 verproductos.php
```

## Insertar Productos

Nombre:

Descripcion:

Categoria:

Precio:

[Ver mis productos](#)

`verproductos.php` es la página que llama a la función (en `cargar.php`) que carga los productos

`insertar.html` es un formulario que se envía a `insertar.php`.

# Excepciones

<?php

```
function sumar($numero1, $numero2)
{
    // verifica si ambos valores ingresados son numéricos.
    if (is_numeric($numero1) && is_numeric($numero2)){
        return $numero1 + $numero2;
    }else{
        // crea una excepción.
        throw new Exception('Los valores ingresados no son numéricos');
        return 0; // comprobar si pasa por aquí. ¿Puede fallar el trigger?
    }
}

try {
    // ejecutamos la función con parámetros incorrectos
    echo sumar('Ab', 'Cd');
} catch(Exception $e) {
    echo $e->getMessage();
}
```

?>



# EJERCICIO

Realiza una copia de tu ejercicio de los **libros** de la semana pasada. Cambia el nombre de la página principal por **ud04ejer04.php** y el título 'Ejercicio 4 - PDO'.

- Modifica todo para **utilizar PDO**. No se puede utilizar **nada de MySQLi**.
- Añade dos **nuevas columnas** en la tabla: [Eliminar](#) y [Modificar](#) , para cada libro.
- Implementa un **buscador** de libros **por título**.
- Controla una **excepción** si las **páginas** insertadas no son un dato numérico.

Id: 1

Título: aaa

Autor: <-- ¡Debes introducir un autor!

Páginas: 111

Introducir libro

ID	TITULO	AUTOR	PAGINAS	ELIMINAR
----	--------	-------	---------	----------

Id:

Título:

Autor:

Páginas:

Introducir libro

ID	TITULO	AUTOR	PAGINAS	ELIMINAR
1	aaa	jajaja	111	<a href="#">Eliminar</a>
2	eee	jejeje	222	<a href="#">Eliminar</a>
3	iii	jijiji	333	<a href="#">Eliminar</a>
4	ooo	jojojo	444	<a href="#">Eliminar</a>
5	uuu	jujuju	555	<a href="#">Eliminar</a>

3	iii	jijiji	333	<a href="#">Eliminar</a>
---	-----	--------	-----	--------------------------



ID	TITULO	AUTOR	PAGINAS	ELIMINAR
1	aaa	jajaja	111	<a href="#">Eliminar</a>
2	eee	jejeje	222	<a href="#">Eliminar</a>
4	ooo	jojojo	444	<a href="#">Eliminar</a>
5	uuu	jujuju	555	<a href="#">Eliminar</a>

[Eliminar](#) es un enlace (<a href= "... ">...</a>)

## Ejemplo de buscador (pero con *productos*).

### Mis Productos

<input type="text"/>					<input type="button" value="BUSCAR"/>	
ID	Nombre	Descripcion	Categorias	Precio		
2	mesa	marron	3	344	<a href="#">Eliminar</a>	<a href="#">Modificar</a>
3	puerta	blanca cristal	4	55	<a href="#">Eliminar</a>	<a href="#">Modificar</a>
4	ventana	blanca cristal	5	120	<a href="#">Eliminar</a>	<a href="#">Modificar</a>
6	caseta	madera roble	5	11000	<a href="#">Eliminar</a>	<a href="#">Modificar</a>
7	ventana	madera	3	230	<a href="#">Eliminar</a>	<a href="#">Modificar</a>

[Nuevo Producto](#)

### Mis Productos

<input type="text" value="ventana"/>					<input type="button" value="BUSCAR"/>	
ventana					Categorias	Precio
2	mesa	marron	3	344	<a href="#">Eliminar</a>	<a href="#">Modificar</a>
3	puerta	blanca cristal	4	55	<a href="#">Eliminar</a>	<a href="#">Modificar</a>
4	ventana1	blanca cristal"	5	120	<a href="#">Eliminar</a>	<a href="#">Modificar</a>
6	caseta	madera roble	5	11000	<a href="#">Eliminar</a>	<a href="#">Modificar</a>
7	ventana	madera	3	230	<a href="#">Eliminar</a>	<a href="#">Modificar</a>

[Nuevo Producto](#)

... LIKE '%ventana%';

### Mis Productos

<input type="text"/>					<input type="button" value="BUSCAR"/>	
ID	Nombre	Descripcion	Categorias	Precio		
4	ventana1	blanca cristal"	5	120	<a href="#">Eliminar</a>	<a href="#">Modificar</a>
7	ventana	madera	3	230	<a href="#">Eliminar</a>	<a href="#">Modificar</a>

[Nuevo Producto](#)

## Ejemplo de modificación (pero con *productos*).

### Modificar Productos

Nombre:

Descripcion:

Categoria:

Precio:

Producto modificado correctamente  
[Ver Productos](#)

### Mis Productos

<input type="text"/>					<input type="button" value="BUSCAR"/>	
ID	Nombre	Descripcion	Categorias	Precio		
2	mesa	marron"	3	360	<a href="#">Eliminar</a>	<a href="#">Modificar</a>
3	puerta	blanca cristal	4	55	<a href="#">Eliminar</a>	<a href="#">Modificar</a>
4	ventana1	blanca cristal"	5	120	<a href="#">Eliminar</a>	<a href="#">Modificar</a>
6	caseta	madera roble	5	11000	<a href="#">Eliminar</a>	<a href="#">Modificar</a>
7	ventana	madera	3	230	<a href="#">Eliminar</a>	<a href="#">Modificar</a>

[Nuevo Producto](#)