ESTABLECER CONEXIONES

Extensión MySQLi

```
$conexion = new mysqli("localhost", "usuario", "contraseña", "bbdd", 3306);
$conexion->set_charset("utf8");
$error = $conexion->connect_errno;
if ($error != null)
{
    print "<p>Se ha producido el error: $conexion->connect_error.</p>";
    exit();
}
else { //hacemos cosas }
$conexion->close();
```

PHP Data Objects (PDO)

```
$opciones = array(PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8");
$conexion = new PDO('mysql:host=localhost;dbname=bbdd', 'usuario',
'contraseña' , $opciones);

unset($conexion);
```

EJECUTAR CONSULTAS

Extensión MySQLi

```
$resultado = $conexion->query('DELETE FROM tabla WHERE condicion');
if ($resultado)
{
	print "<p>Se han borrado $conexion->affected_rows registros.</p>";
}
```

PHP Data Objects (PDO)

```
$registros = $conexion->exec('DELETE FROM tabla WHERE condicion');
print "<p>Se han borrado $registros registros.</p>";
```

TRANSACCIONES

¡Deberán estar soportadas por el motor!

Extensión MySQLi

```
$todoOk = true;  // Definimos una variable para comprobar la ejecución
$conexion->autocommit(false); // Deshabilitamos el modo transaccional
automático

$sql = 'UPDATE tabla SET campo=1 WHERE condicion';
if ($conexion->query($sql) != true) $todoOk = false;  //Si hay error ponemos
false

$sql = 'INSERT INTO tabla (`campo1`, `campo2`) VALUES ("valor1", "valor2")';
if ($conexion->query($sql) != true) $todoOk = false; //Si hay error ponemos
false

// Si todo fue bien, confirmamos los cambios y en caso contrario los
deshacemos
if ($todoOk == true)
{
	$conexion->commit();
	print "<p>Los cambios se han realizado correctamente.</p>";
}
else
 {
	$conexion->rollback();
	print "<p>No se han podido realizar los cambios.</p>";
}
```

PHP Data Objects (PDO)

```
$todoOk = true; // Definimos una variable para comprobar la ejecución
$conexion->beginTransaction();// Iniciamos la transacción

$sql = 'UPDATE tabla SET campo=1 WHERE condicion';
if ($conexion->exec($sql) == 0) $todoOk = false; //Si hay error ponemos false

$sql = 'INSERT INTO tabla (`campo1`, `campo2`) VALUES ("valor1", "valor2")';
if ($conexion->exec($sql) == 0) $todoOk = false; //Si hay error ponemos false

// Si todo fue bien, confirmamos los cambios  y en caso contrario los
deshacemos
if ($todoOk == true)
{
	$conexion->commit();
	print "<p>Los cambios se han realizado correctamente.</p>";
}
else
{
	$conexion->rollback();
	print "<p>No se han podido realizar los cambios.</p>";
}
```

OBTENCIÓN Y UTILIZACIÓN DE CONJUNTOS DE RESULTADOS

Extensión MySQLi

Para acceder al primer campo devuelto de la select:

```
$resultado = $conexion->query('SELECT campo1, campo2 FROM tabla WHERE
condicion');
$registro = $resultado->fetch_array();  // Obtenemos el primer registro
$campo1 = $registro['campo1']; // O también $registro[0];
$campo2 = $registro['campo2']; // O también $registro[1];
print "Mostramos el $campo1 y  $campo2";
```

fetch_array devuelve un array que contiene tanto claves numéricas como asociativas.
Este comportamiento por defecto se puede modificar utilizando un parámetro opcional, que puede tomar los siguientes valores:

- **MYSQLI_NUM**. Devuelve un array con claves numéricas.
- **MYSQLI_ASSOC**. Devuelve un array asociativo.
- **MYSQLI_BOTH**. Es el comportamiento por defecto, en el que devuelve un array con claves numéricas y asociativas.

Existen otros métodos para trabajar con los datos obtenidos en vez de utilizar **fetch_array**:
fetch_assoc: Idéntico a `fetch_array` pasando como parámetro **MYSQLI_ASSOC**.
fetch_row: Idéntico a `fetch_array` pasando como parámetro **MYSQLI_NUM**.
fetch_object: Similar a los métodos anteriores, pero devuelve un objeto en lugar de un array. Las propiedades del objeto devuelto se corresponden con cada uno de los campos del registro.

Para recorrer todos los registros podemos utilizar el método fetch_object por ejemplo:

```
$resultado = $conexion->query('SELECT campo1, campo2 FROM tabla WHERE
condicion');
$objetoDatos = $resultado->fetch_object();
while ($objetoDatos!= null)
{
	print "<p>Los campos son $objetoDatos->campos1: $objetoDatos->campo2
	</p>";
	$objetoDatos = $resultado->fetch_object();
}
```

PHP Data Objects (PDO)

Al igual que en MySQLi, en PDO hay varias posibilidades para tratar con el conjunto de resultados devueltos por el método **query**. La más utilizada es el método **fetch**:

```
$resultado = $conexion->query('SELECT campo1, campo2 FROM tabla WHERE
condicion');
while ($registro = $resultado->fetch())
{
	echo $registro['campo1'].": ".$registro['campo2']."<br />";
}
```

Por defecto, el método **fetch** genera y devuelve, a partir de cada registro, 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 solo un array asociativo.
- **PDO::FETCH_NUM**. Devuelve solo un array con claves numéricas.
- **PDO::FETCH_BOTH**. Devuelve un array con claves numéricas y asociativas. Es el comportamiento por defecto.
- **PDO::FETCH_OBJ**. Devuelve un objeto cuyas propiedades se corresponden con los campos del registro.

```
$resultado = $conexion->query('SELECT campo1, campo2 FROM tabla WHERE
condicion');
while ($registro = $resultado->fetch(PDO::FETCH_OBJ))
{
	echo $registro->campo1.": ".$registro->campo2."<br />";
}
```

- **PDO::FETCH_LAZY**. Devuelve tanto el objeto como el array con clave dual anterior.
- **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.

```
$resultado = $conexion->query('SELECT campo1, campo2 FROM tabla WHERE
condicion');
$resultado->bindColumn(1, $campo1);
$resultado->bindColumn(2, $campo2);
while ($registro = $resultado->fetch(PDO::FETCH_BOUND))
{
	echo $campo1.": ".$campo2."<br />";
}
```

CONSULTAS PREPARADAS

Extensión MySQLi

```
$consulta = $conexion->stmt_init();
$consulta->prepare('INSERT INTO tabla (campo1, campo2) VALUES (?, ?)');
$campo1 = "valor_campo1";
$campo2 = "valor_campo2";
$consulta->bind_param('ss', $campo1, $campo2);
$consulta->execute();
$consulta->close();
```

Antes de ejecutar la consulta tienes que utilizar el método **`bind_param`** (o la función **`mysqli_stmt_bind_param`**) para sustituir cada parámetro por su valor. El primer parámetro del método `bind_param` es una cadena de texto en la que cada carácter indica el tipo de un parámetro, según la siguiente tabla:

Caracteres indicativos del tipo de los parámetros en una consulta preparada	
Carácter	**Tipo del parámetro**
i	Número entero
d	Número real
s	Cadena de texto
b	Contenido el formato binario

En el caso de las consultas que devuelven valores, se puede utilizar el método **`bind_result`**:

```
$consulta = $conexion->stmt_init();
$consulta->prepare('SELECT campo1, campo2 FROM tabla WHERE condicion');
$consulta->execute();
$consulta->bind_result($campo1, $campo2);
while($consulta->fetch())
{
	echo $campo1.": ".$campo2."<br />";
}
$consulta->close();
```

PHP Data Objects (PDO)

```
$consulta=$conexion->prepare('INSERT INTO tabla (campo1, campo2) VALUES (?,
?)');
```

Se pueden utilizar parámetros con nombre, precediéndolos por el símbolo de dos puntos:

```
$consulta=$conexion->prepare('INSERT INTO tabla (campo1, campo2) VALUES (:cm1,
:cm2)');

$campo1 = "valor_campo1";
$campo2 = "valor_campo2";
$consulta->bindParam(1, $campo1); //o $consulta->bindParam(:cm1, $campo1);
$consulta->bindParam(2, $campo2); //o $consulta->bindParam(:cm2, $campo2);
$consulta->execute();
```

Si se quieren devolver valores podemos hacer algo así:

```
$consulta = $conexion->prepare('SELECT * FROM tabla WHERE campo = ?');
$consulta->bindParam(1, $campo1);
if ($consulta->execute())
{
    while ($fila = $consulta->fetch())
    {
   $datos[] = array("valor1" => $fila['valor1'], "valor2" => $fila['valor2']);
    }
}
unset($conexion);
return $datos;
```