

La clase MySQLi

Accediendo a bases de datos

Sentencias preparadas

Sentencias preparadas

- Una de las ventajas que aporta el uso de la clase **mysqli** frente al estilo procedimental son las **sentencias preparadas**
- Una sentencia preparada es aquella en la que el código de la sentencia incluirá ciertos **parámetros** que se dejan para rellenar más adelante.
- De esta forma, la sentencia se podrá ejecutar de la misma forma o muy similar de manera muy rápida y eficiente.

Sentencias preparadas: ventajas

- **Reducen** el tiempo de análisis ya que la preparación de la consulta se hace una sola vez.
- Son muy útiles **contra inyecciones** ya que los valores de los parámetro se transmiten después usando un protocolo diferente y no necesitan ser escapados.
- La ventaja que tienen las sentencias preparadas es que podemos incluir en ellas **parámetros** que se rellenarán al momento de ejecutarlas.
- Los parámetros se indican con **?**

Sentencias preparadas

- La ejecución de una sentencia preparada tiene dos fases:
 - **Preparación y comprobación:** se crea la plantilla de la sentencia y el servidor comprueba que no hay errores de sintaxis.
 - **Vinculación y ejecución:** se rellenan los parámetros y se ejecuta la sentencia.

Sentencias preparadas

- **mysqli::prepare(sentencia):**

- Prepara una sentencia para su posterior ejecución.

```
$sentencia = "select * from alumnos where nombre = ?";  
$consulta = $conexion->prepare($sentencia);
```

- La llamada a «prepare» envía la plantilla de la consulta a la base de datos.
- La base de datos analiza, compila y optimiza la consulta y la guarda **sin ejecutar**.

Sentencias preparadas

- Los pasos que seguiremos siempre serán:
 1. Preparar la consulta con los parámetros necesarios.
 2. Asignar variables para almacenar los datos devueltos por la consulta
 3. Asignar valores a los parámetros de la consulta
 4. Ejecutar la consulta
 5. Sacar los datos devueltos por la consulta

Sentencias preparadas

- El método `mysqli::prepare()` devuelve un objeto de la clase **`mysqli_stmt`** que representa a la sentencia preparada.

La clase mysqli_stmt

mysqli_stmt: Propiedades

- ✓ **mysqli_stmt::affected_rows**: número de filas afectadas por una sentencia **no select**
- ✓ **mysqli_stmt::num_rows**: número de filas devueltas por la sentencia **select**
- ✓ **mysqli_stmt::errno**: código del error ocurrido
- ✓ **mysqli_stmt::error**: texto descriptivo del error
- ✓ **mysqli_stmt::field_count**: número de columnas devueltas por la sentencia **select**
- ✓ **mysqli_stmt::param_count**: número de parámetros de la sentencia

mysqli_stmt: métodos

- **mysqli_stmt::bind_param(tipos, list)**

- ✓ Permite asignar valores a los parámetros de la sentencia.
- ✓ Recibe dos elementos
 - ✓ El tipo de dato de los valores que va a recibir:
 - ✓ i => si el valor asignado es un número sin decimales
 - ✓ d => si el valor asignado es un número con decimales
 - ✓ s => si el valor asignado es una cadena de caracteres
 - ✓ Listado con los valores que recibe para asignar a los parámetros

mysqli_stmt: métodos

```
$sentencia = "select * from alumnos where nombre = ?";  
$consulta = $conexion->prepare($sentencia);  
  
$nom = $_POST['nombre'];  
  
$consulta->bind_param("s", $nom);
```

- Si la sentencia preparada tiene más de un parámetro:
 - ✓ El tipo se indica en una única cadena de texto
 - ✓ los valores se separan por comas, en orden correcto

```
$sentencia = "select * from alumnos where nombre = ? and edad = ?";  
$consulta = $conexion->prepare($sentencia);  
  
$nom = $_POST['nombre'];  
$ed = $_POST['edad'];  
  
$consulta->bind_param("si", $nom, $ed);
```

mysqli_stmt: métodos

- **mysqli_stmt::bind_result(lista)**

- ✓ Permite asignar variables para recoger los valores devueltos por un select.
- ✓ Se asignará una variable por cada columna del select

```
$nom = $_POST['nombre'];  
$ed = $_POST['edad'];  
  
$sentencia = "select * from alumnos where nombre = ? and edad = ?";  
$consulta = $conexion->prepare($sentencia);  
  
$consulta->bind_param("si", $nom, $ed);  
  
$consulta->bind_result($dni, $nombre, $edad);
```

mysqli_stmt: métodos

- **mysqli_stmt::execute()**

- ✓ Ejecuta una sentencia preparada anteriormente
- ✓ Buscará los valores que haya en las variables asociadas a la sentencia.

```
$sentencia = "select * from alumnos where nombre = ? and edad = ?";  
$consulta = $conexion->prepare($sentencia);  
.....  
$consulta->bind_param("si", $nom, $ed);  
  
$consulta->bind_result($dni, $nombre, $edad);  
  
$consulta->execute();
```

mysqli_stmt: métodos

- **mysqli_stmt::fetch()**

- ✓ Extrae una fila de los valores devueltos por la consulta
- ✓ Volcará el resultado en las variables que se asociaron.
- ✓ Esas variables ya se pueden usar para lo que sea necesario
- ✓ Devuelve FALSE si no hay datos que sacar

```
$sentencia = "select * from alumnos where nombre = ? and edad = ?";
$consulta = $conexion->prepare($sentencia);

...

$consulta->bind_param("si", $nom, $ed);
$consulta->bind_result($dni, $nombre, $edad);
$consulta->execute();

echo "Mostrando los alumnos buscados <br>";
while($consulta->fetch())
{
    echo "- $dni, $nombre, $edad <br>";
}
```

mysqli_stmt: métodos

- **mysqli_stmt::close()**

- ✓ Cierra la sentencia preparada y desvincula las variables usadas como parámetros o como resultado

```
$sentencia = "select * from alumnos where nombre = ? and edad = ?";
$consulta = $conexion->prepare($sentencia);

...

$consulta->bind_param("si", $nom, $ed);
$consulta->bind_result($dni, $nombre, $edad);
$consulta->execute();

echo "Mostrando los alumnos buscados <br>";
while($consulta->fetch())
{
    echo "- $dni, $nombre, $edad <br>";
}

$consulta->close();
$conexion->close();
```


Ejemplos

Sentencias preparadas: Ejemplo 1

- Mostrar los alumnos con una edad inferior a...

```
<?php
$conexion = new mysqli('localhost', 'root', '', 'centro');
//preparo la sentencia
$sentencia = $conexion->prepare("select * from alumnos where edad <= ?");
$edad_buscar = 21;
//Asigno variables para almacenar valores
$sentencia->bind_result($dni, $nombre, $edad);
//asigno los parámetros
$sentencia->bind_param("i", $edad_buscar);
//Ejecuto la sentencia
$sentencia->execute();
//Almaceno los resultados
$sentencia->store_result();
//Muestro los resultados
echo "Hay un total de $sentencia->affected_rows alumnos para mostrar: ";
while($sentencia->fetch())
{
    echo "<br>Alumno: $dni, $nombre, $edad";
}
//Cierro todo
$sentencia->close();
$conexion->close();
```

?>

Sentencias preparadas: Ejemplo 2

Mostrar los alumnos con 21 o 24 años...

```
$conexion = new mysqli('localhost', 'root', '', 'centro');
$conexion->set_charset("utf8");
//preparo la sentencia
$sentencia = $conexion->prepare("select * from alumnos where edad = ? or edad = ?");
//Asigno variables para almacenar valores
$sentencia->bind_result($dni, $nombre, $edad);
//asigno los parámetros
$edad1=21;
$edad2=24;
$sentencia->bind_param("ii", $edad1, $edad2);
// //Ejecuto la sentencia
$sentencia->execute();
// //Almaceno los resultados
$sentencia->store_result();
// //Muestro los resultados
echo "Hay un total de $sentencia->affected_rows alumnos para mostrar: ";
while($sentencia->fetch())
{
    echo "<br>Alumno: $dni, $nombre, $edad";
}
//Cierro todo
$sentencia->close();
$conexion->close();
```

Sentencias preparadas: Ejemplo 3

Mostrar los alumnos cuyo nombre contenga...

```
<?php
    $conexion = new mysqli('localhost', 'root', '', 'centro');
    $conexion->set_charset("utf8");
    //preparo la sentencia
    $sentencia = $conexion->prepare("select * from alumnos where nombre like ?");
    //Asigno variables para almacenar valores
    $sentencia->bind_result($dni, $nombre, $edad);
    //asigno los parámetros
    $nombre_buscar="%a%";
    $sentencia->bind_param("s", $nombre_buscar);
    //$sentencia->bind_param(":valor", $nombre_buscar);
    // //Ejecuto la sentencia
    $sentencia->execute();
    // //Almaceno los resultados
    $sentencia->store_result();
    // //Muestro los resultados
    echo "Hay un total de $sentencia->affected_rows alumnos para mostrar: ";
    while($sentencia->fetch())
    {
        ..... echo "<br>Alumno: $dni, $nombre, $edad";
    }
    //Cierro todo
    $sentencia->close();
    $conexion->close();
?>
```

Sentencias preparadas: ventajas

- Como se ha dicho antes, no será necesario volver a compilar una sentencia que ya esté preparada
- El propio MySQL volverá a buscar la variables asociadas para la siguiente ejecución

```
$conexion = new mysqli('localhost', 'root', '', 'centro');
$conexion->set_charset('utf8');

$nom = 'Paloma Ruiz';
$ed = 24;

$sentencia = "select * from alumnos where nombre = ? and edad = ?";
$consulta = $conexion->prepare($sentencia);

$consulta->bind_param("si", $nom, $ed);
$consulta->bind_result($dni, $nombre, $edad);

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

echo "El primer alumno buscado tiene los siguientes datos: $dni, $nombre, $edad";

$nom = 'Isabel Perea';
$ed = 25;

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

echo "<br>El segundo alumno buscado tiene los siguientes datos: $dni, $nombre, $edad";

$consulta->close();
$conexion->close();
```

Sentencias preparadas: ventajas

- No sólo vamos a utilizar las sentencias preparadas para mostrar valores
- También se pueden usar para insertar, borrar o actualizar.
- En el siguiente ejemplo vamos a insertar un nuevo alumno:

```

//primero recogemos los datos del formulario
$dni=$_POST['dni'];
$nombre = $_POST['nombre'];
$edad = $_POST['edad'];

//creamos el objeto tipo conexión
$conexion = new mysqli("localhost", "root", "", "centro");

//comprobamos que no haya ningún alumno ya con esos dato insertado
$compruebaAlumno = "select count(dni) num from alumnos where dni = ? and nombre = ? and edad = ?";
//preparamos la consulta
$consulta=$conexion->prepare($compruebaAlumno);
//asignamos contenido a las ? de la consulta
$consulta->bind_param("ssi", $dni, $nombre, $edad);
//ejecutamos la consulta
$consulta->execute();
//Comprobamos cuantas filas ha devuelto
//asigno variable para meter cantidad
$consulta->bind_result($num);
$consulta->fetch();
if($num == 0)
{
    //como ya no voy a usar más la consulta de antes, la puedo cerrar
    $consulta->close();
    //como no hay nadie con esos datos, puedo insertar
    $insertar = "insert into alumnos values (?, ?, ?)";
    $consulta_inserta = $conexion->prepare($insertar);
    $consulta_inserta->bind_param("ssi", $dni, $nombre, $edad);
    $consulta_inserta->execute();
    //ya está insertado, puedo cerrar
    $consulta_inserta->close();
}
else
{
    echo "Ya hay un alumno con esos datos";
}
$conexion->close();

```


La clase `mysqli_stmt`

Resumen de características y métodos

La clase mysqli_stmt: Características

Atributo	Tipo	Descripción
\$affected_rows	int	Almacena el número de filas afectadas por la ultima sentencia ejecutada (no Select)
\$errno	int	Devuelve el código de error producido por la última sentencia ejecutada
\$error_list	array	Devuelve una lista de errores producidos por la última sentencia ejecutada
\$error	string	Devuelve una cadena con el error producido por la última sentencia ejecutada
\$field_count	int	Número de campos de la sentencia dada
\$insert_id	int	Devuelve el id generado en la última operación INSERT
\$num_rows	int	Devuelve el número de filas resultado de ejecutar un select
\$param_count	int	Número de parámetros de la sentencia dada

La clase mysqli_stmt: métodos

Método	Devuelve	Descripción
attr_get (\$at)	int	Devuelve el valor actual de un atributo de la sentencia
bind_param(list)	bool	Agrega variables a la sentencia
bind_result(list)	bool	Vincula variables a la sentencia para almacenar resultados
data_seek(int \$n)	void	Busca una fila concreta dentro del resultado
execute()	Bool	Ejecuta la sentencia preparada
fetch()	Bool	Obtiene una fila del resultado
free_result()	Void	Libera la memoria de los resultados almacenadas
get_result()	mysqli_result	Obtiene el resultado de ejecutar una sentencia
prepare(\$consulta)	mixed	Prepara la sentencia
close()	bool	Cierra una sentencia preparada
reset()	bool	Reinicia una sentencia preparada

Ejemplo uso de inyección SQL

Veremos ahora cómo las sentencias preparadas nos protegen contra inyecciones SQL

Control de acceso de usuarios

- Vamos a suponer una base de datos con una tabla para los usuarios.
- La base de datos se llamará “bd” y la tabla usuarios debe ser como la siguiente:

id	nombre	salario	pass
0	admin	0.00	admin
1	Juan	444.00	juan
2	Pedro	3256.20	pedro
3	Maria	4522.00	maria
4	Ana	325.20	ana
5	Pepe	152.00	pepe

Control de acceso de usuarios

- En una aplicación normal, el usuario se va a encontrar un formulario de acceso.
- Algo parecido a esto:

```
<form action='#' method='post'>
  nombre de usuario <input type='text' name='nombre'>
  <br>
  contraseña <input type='text' name='pass'>
  <br>
  <input type='submit' name='enviar'>
</form>
```

Control de acceso de usuarios

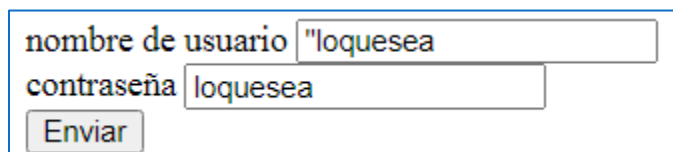
- Si queremos controlar el acceso sin utilizar sentencias preparadas, el proceso sería como esto:

```
$usuario = $_POST['nombre'];
$pass = $_POST['pass'];

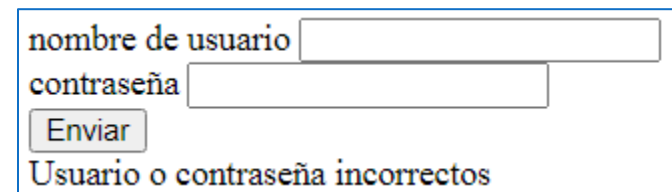
$conexion = new mysqli('localhost', 'root', '', 'bd');
$sentencia = "select * from usuarios where nombre='$usuario' and pass='$pass'";
$resultado=$conexion->query($sentencia);
if($resultado)
{
    if($resultado->num_rows > 0)
    {
        echo "estas dentro";
    }
    else
    {
        echo "Usuario o contraseña incorrectos";
    }
}
else
{
    echo "error en la consulta";
}
```

Control de acceso de usuarios

- Vamos a comprobar si nuestro control es seguro.



A screenshot of a web login form. It has two input fields: 'nombre de usuario' with the value 'loquesea' and 'contraseña' with the value 'loquesea'. Below the fields is a button labeled 'Enviar'.



A screenshot of a web login form, similar to the previous one, but with an error message. The 'nombre de usuario' and 'contraseña' fields are empty. Below the fields is a button labeled 'Enviar'. Below the button, the text 'Usuario o contraseña incorrectos' is displayed.

- Al conseguir ese mensaje ya sabemos que la aplicación utiliza los datos enviados sin tratamiento previo.
- Ha dado error porque la consulta se ha convertido en algo así:

```
select * from usuarios where nombre="'loquesea' and pass='loquesea'
```

- No hay errores de sintaxis, pero no existe el usuario con unas “al principio de su nombre

Control de acceso de usuarios

- Probamos ahora con comilla simple

nombre de usuario

contraseña

nombre de usuario

contraseña

error en la consulta

- Ahora ya sabemos que la consulta está delimitada por comillas dobles porque se ha convertido en algo así:

```
select * from usuarios where nombre="'loquesea' and pass='loquesea'
```

- En este caso ya sí hay error en la consulta porque cerramos las comillas antes del texto

Control de acceso de usuarios

- Con esta información, ya podemos intentar engañar a la aplicación introduciendo una cadena de texto que modifique la consulta y nos permita entrar:

nombre de usuario	<input type="text" value="loquesea"/>
contraseña	<input type="text" value="loquesea' OR '1'='1"/>
<input type="button" value="Enviar"/>	

Control de acceso de usuarios

- Ahora la consulta es:

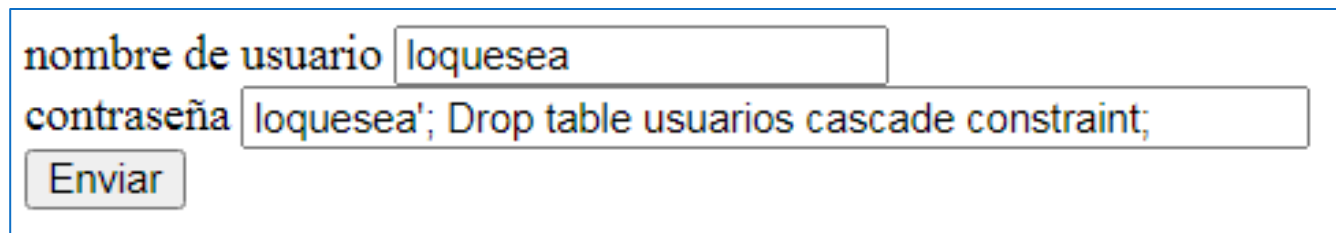
```
select * from usuarios where nombre='loquesea' and pass='loquesea' OR '1'='1'
```

- Y ese $1=1$ siempre será verdadero, así que siempre entraré.
- Es más, como el primer usuario de la base de datos suele ser el administrador,

¡¡¡entraré con perfil de administrador!!!

Control de acceso de usuarios

- ¿Qué pasaría si relleno el formulario de esta forma?



A screenshot of a web form with two input fields and a submit button. The first field is labeled 'nombre de usuario' and contains the text 'loquesea'. The second field is labeled 'contraseña' and contains the text 'loquesea'; Drop table usuarios cascade constraint;'. Below the fields is a button labeled 'Enviar'.

nombre de usuario	loquesea
contraseña	loquesea'; Drop table usuarios cascade constraint;
<input type="button" value="Enviar"/>	

- No podría entrar, pero...
¡borraría toda la tabla de los usuarios!

Control de acceso de usuarios

- Partiendo del mismo formulario, vamos ahora a hacer el control de los datos utilizando sentencias preparadas.
- El código en este caso quedaría así:

Control de acceso de usuarios

```
$conexion = new mysqli('localhost', 'root', '', 'bd');
$sentencia="select id from usuarios where nombre=? and pass=?";

$consulta = $conexion->prepare($sentencia);
$consulta->bind_param("ss", $usuario, $pass);
$consulta->bind_result($id);

if($consulta->execute())
{
    $consulta->fetch();
    if(isset($id))
    {
        echo "estas dentro";
    }
    else
    {
        echo "Usuario o contraseña incorrectos";
    }
}
else
{
    echo "error en la consulta";
}
```

Control de acceso de usuarios

- En este caso, la consulta no lleva comillas ni nada que pueda permitir modificar el código de la propia consulta.
- Podemos probar todas las opciones anteriores y todas ellas nos darán el mismo resultado: **no tendremos acceso**
- Para más ejemplos de inyecciones SQL:

<https://www.mclibre.org/consultar/php/lecciones/php-db-inyeccion-sql.html>

La clase mysqli

Accediendo a bases de datos