Tema 5: Utilización de técnicas de acceso a datos

¿Qué aprenderás?

- Cómo conectarnos a una base de datos MySQL desde nuestra aplicación web en PHP.
- Cómo realizar las operaciones típicas sobre una base de datos: inserción, modificación, borrado y consulta de datos.

¿Sabías que…?

- Para el trabajo con bases de datos MySQL, PHP incorpora la implementación mysql, que está obsoleta a partir de la versión PHP 5.5. En su lugar, hay que usar la implementación mysqli.
- PHP también incorpora funciones para trabajar con otras bases de datos, como Oracle, SQL Server o Access.

1. Conexión a una base de datos MySQL

1.1. Introducción

Una de las principales funcionalidades que presenta PHP es su interactuación con bases de datos. PHP proporciona al programador funciones para comunicarse con una base de datos MySQL y gestionar los datos que hay en ella.

Todas las funciones referentes a la gestión de una base de datos MySQL tienen un mismo formato. No obstante, dependiendo de la versión de PHP con la que se trabaje (anterior a la versión 5.5 o posterior), este formato será diferente:

```
mysql_funcion (parámetro1, parámetro2, ...);
mysqli_funcion (parámetro1, parámetro2, ...);
```

Como se observa, para versión de PHP a partir de la 5.5.0 (incluida) los nombres de las funciones tienen el prefijo "mysqli_". Se ha añadiendo una "i" a los nombres de las mismas funciones de las versiones anteriores a 5.5. Esta "i" significa improved (mejorado).

En la actualidad coexisten ambos formatos de funciones, y se pueden usar unas u otras indistintamente, ya que son equivalentes (hacen lo mismo). No obstante, se recomienda el uso de las funciones más actuales, ya que las anteriores se eliminarán en futuras especificaciones de PHP.

1.2. Conectarse al servidor MySQL

Lo primero que tenemos que hacer para trabajar con una base de datos es conectarnos a ella. Para ello necesitamos saber es el nombre del PC donde se encuentra la base de datos, los datos de la cuenta para acceder a MySQL (usuario y contraseña), y el nombre de la base de datos con la que queremos conectar.

Abrimos la conexión con el servidor MySQL con la siguiente función:

```
$conexion = mysqli_connect(host, username, password,
dbname, port, socket);
```

Los parámetros de la función mysqli_connect son los siguientes:

- host: el nombre o IP del PC donde MySQL está instalado. Si la base de datos está en el mismo PC que el sitio web se pueden usar los valores NULL o localhost.
- username: el nombre de usuario de MySQL.
- password: la contraseña del usuario anterior. Si el usuario no tiene contraseña, este parámetro se deja en blanco ("").
- dbname: el nombre de la base de datos a utilizar para realizar consultas. Es un parámetro opcional, puede no especificarse.
- port: especifica el número de puerto al que intentar conectar al servidor MySQL. Si no se especifica, usará el puerto por defecto para conectarse a MySQL: el 3306.

- socket: especifica el socket que debería usarse para la conexión. Es opcional.

Un ejemplo de conexión con una base de datos sería:

```
$conexion = mysqli_connect("localhost", "root", "12345",
"Empresa") or die ("Error en la conexión a la BD);
```

En el código anterior intentamos abrir una conexión a la base de datos Empresa, situada en la misma máquina en que ejecutamos el código PHP. Nos autenticamos con el usuario root y la contraseña 12345. Si la conexión tiene éxito, se almacenarán los datos de esta en la variable $conexion. Si la conexión falla, el programa se detiene en ese punto y envía el mensaje "Error en la conexión a la BD" al explorador.

1.3. Seguridad en la conexión

Por razones de seguridad es buena idea almacenar la información de la conexión en variables y usarlas en el enunciado de conexión:

```
$servidor = "localhost";
$usuario = "root";
$pass = "12345";
$db = "Empresa";
$conexion = mysqli_connect($servidor, $usuario, $pass,
$db) or die ("Error en la conexión a la BD);
```

Otra forma de hacer la conexión más segura, es poner los enunciados de asignación para los datos sobre la conexión en un archivo separado, en un lugar oculto, de modo que el nombre de la cuenta y la contraseña no estén ni siquiera en el programa. Lo único que tendremos que hacer para usar estos datos será incluirlos en el/los archivo/s en el/los que hagamos la conexión mediante el enunciado include.

1.4. Cerrar una conexión al servidor MySQL

PHP nos permite mantener abiertas tantas conexiones sobre un servidor MySQL como deseemos. Si la conexión la hacemos con código del apartado anterior, la variable $conexion contiene información que identifica la conexión.

Cerraremos una conexión de la siguiente forma:

```
mysqli_close($conexion);
```

El código anterior cierra la conexión cuyos datos se guardaban en la variable $conexion. Le función mysqli_close devuelve TRUE en caso de éxito o FALSE en caso de error.

1.5. Seleccionar una base de datos

Pese que las versiones posteriores a la 5 de PHP proporcionan la función mysqli_connect, que ofrece la posibilidad de seleccionar la base de datos con la que queremos conectar mediante un parámetro, este parámetro es opcional, puede no especificarse. De esta forma,

habremos establecido y abierto la conexión con el servidor MySQL. Ahora necesitamos decir la base de datos con la cuál queremos interactuar. Lo hacemos con la función:

```
mysqli_select_db ($conexion, "nombre_base_datos");
```

En el siguiente ejemplo conectamos con el servidor MySQL y seleccionamos la base de datos "Empresa":

```
$servidor = "localhost";
$usuario = "root";
$pass = "12345";
$conexion = mysqli_connect($servidor, $usuario, $pass) or
die ("Error en la conexión al servidor);
$base_datos = "Empresa";
$db = mysqli_select_db ($conexion, $base_datos) or die
("Error en la conexión a la base de datos");
```

Podemos usar la función mysqli_select_db para cambiar la base de datos con la que estamos trabajando sin necesidad de crear otra conexión con el servidor.

1.6. Control de errores en MySQL

Las funciones mysql que se usan en PHP pueden enviar mensajes de error MySQL con información sobre los problemas que hayan podido surgir. Estos mensajes no serán enviados al explorador a menos que el programa lo mande deliberadamente.

Hay tres formas comunes de llamar a las funciones mysql:

- Llamar a la función sin manejo de errores: por ejemplo, podemos abrir una conexión a un servidor MySQL con el siguiente código:

  ```
  $conexion = mysqli_connect($servidor, $usuario,
  $pass);
  ```

  Si se produce un error la conexión no se hace, pero el programa continua ejecutándose, con lo que es probable que se intenten ejecutar instrucciones que dependan de la conexión con MySQL.

- Llamar a la función con un enunciado die: la llamada a la función va acompañada por el método die, que se ejecuta si se produce algún error en MySQL, envía el mensaje asociado y el programa se detiene. Un ejemplo sería:

  ```
  $conexion = mysqli_connect($servidor, $usuario,
  $pass) or die ("Error en la conexión al servidor);
  ```

- Llamar a la función en un bloque if: la función es llamada usando el bloque if que ejecuta las instrucciones asociadas si la conexión falla. Por ejemplo:

  ```
  if(!$conexion = mysqli_connect($servidor, $usuario,
  $pass))
  {
      $mensaje = mysqli_error();
      echo $mensaje;
      die();
  ```

```
}
```

La función mysqli_error devuelve el mensaje de error de la llamada más reciente a una función mysql que haya fallado. El mensaje de error lo guardamos en la variable $mensaje, para mostrarla por el explorador mediante el enunciado echo. El programa termina con la instrucción die. Obsérvese el signo de admiración ! delante de la condición del bloque if. Es el operador de negación (not). Por tanto, el bloque de instrucciones asociadas al if se ejecutarán si la conexión no (not) es cierta.

En nuestros programas debemos hacer siempre un control de errores. En la fase de desarrollo, es recomendable utilizar el tercer método de control de errores, ya que la función mysqli_error nos ofrece información detallada del problema ocurrido. Por ejemplo, si el error se produce a la hora de conectar con el servidor MySQL debido a que el usuario lo hemos introducido mal, veremos este mensaje de error:

```
Access denied for user: 'root'@'localhost' (using
password YES)
```

Este mensaje técnico nos informa del error producido, y podemos ver el usuario y el servidor al que queremos acceder. Además, nos informa de que estamos usando contraseña.

Este tipo de mensajes van bien en fase de desarrollo de nuestra aplicación, pero no está bien que sean vistos por los clientes. En la fase de explotación, será mejor usar el control de errores con el enunciado die, para mostrar los mensajes que nosotros queramos.

2. Realizar consultas sobre la base de datos

2.1. Introducción

Las consultas u operaciones de selección son las que se usan más frecuentemente cuando se trabaja con una base de datos. Lo que más veces querremos hacer será ver un listado de los datos que guarda la base de datos.

En este capítulo trataremos las operaciones que provee PHP para obtener los datos almacenados en una base de datos e interactuar con ellos en nuestra aplicación web.

2.2. Ejecución de sentencias SQL

Una vez hemos abierto la conexión con una base de datos, lo siguiente que haremos será enviar operaciones SQL, ya sea para almacenar, actualizar o recuperar algunos datos.

Para ejecutar sentencias SQL en PHP usaremos la función mysqli_query. Esta función usa dos parámetros. El primero es la variable que guarda los datos de la conexión (recordemos la función mysqli_connect). El segundo es una cadena de caracteres que contiene la sintaxis de la operación SQL que queramos ejecutar. Veamos un ejemplo de uso:

```
$con = mysqli_connect("localhost", "root", "12345",
"Escuela") or die ("Error en la conexión");
$consulta = "SELECT * FROM Alumno";
$resultado = mysqli_query($con, $consulta) or die("Error
en la ejecución de la consulta");
```

En el fragmento de código anterior se puede ver que se crea la variable $consulta para almacenar la consulta SQL que queremos ejecutar en la base de datos. Esta variable se envía como parámetro a la función mysqli_query, que devolverá un valor que se guardará en la variable $resultado.

La función mysqli_query puede utilizarse para enviar a la base de datos operaciones SQL como son SELECT, INSERT, UPDATE o DELETE. El resultado devuelto será:

- TRUE si se ejecutan exitosamente operaciones SQL que no obtienen datos, como INSERT, UPDATE o DELETE. Si se produce algún error en la operación (debido, por ejemplo, a que la sintaxis es errónea) la función devolverá FALSE.
- Si ejecutamos operaciones que devuelven datos, como SELECT, se generará un identificador de donde podremos obtener los datos devueltos por la instrucción SELECT. Si la operación falla, obtendremos el valor FALSE.

Este segundo punto puede resultar un poco confuso. Hay que tener en cuenta el proceso a seguir para obtener datos almacenados en una base de datos:

1. Construimos la sentencia SELECT y la enviamos a la base de datos. Cuando la consulta se ejecuta, los datos seleccionados se almacenen en una ubicación temporal.
2. Movemos los datos de la ubicación temporal a las variables de nuestro programa.

El primer punto hemos visto que lo podemos hacer con la función mysqli_query. Veamos cómo realizar el segundo punto.

2.3. Extraer los datos obtenidos de una consulta

Para trabajar con los datos obtenidos por una sentencia SELECT usaremos la función mysqli_fetch_array.

Después de ejecutar la consulta con mysqli_query, recordemos que los datos estarán en una ubicación temporal. La función mysqli_fetch_array extrae una fila de datos de esta ubicación temporal.

En el siguiente fragmento de código se puede ver un ejemplo de uso de la función mysqli_fetch_array:

```
$consulta = "SELECT * FROM Usuario WHERE user='admin' AND
pass='12345'";
$resultado = mysqli_query($con, $consulta)
     or die("Error en la ejecución de la consulta");
$fila = mysqli_fetch_array($resultado);
if($fila){
     echo "Bienvenido ".$fila['nombre'];
}
```

```
else{
     echo "El usuario no existe";
}
```

El código anterior realiza una consulta a la base de datos sobre la tabla Usuario, con unos valores determinados para los campos user y pass. El resultado de la ejecución de la consulta se almacena en la variable $resultado. A continuación usamos la función mysqli_fetch_array para acceder a la primera fila de los datos devueltos, si los hay. De hecho, el if mira precisamente eso, si la consulta ha devuelto alguna fila. Si el resultado de la consulta no devuelve datos, la variable $fila valdrá NULL, y se ejecutará el bloque de instrucciones del else.

Obsérvese que el valor devuelto por la función mysqli_fetch_array es un array que contiene los valores de la primera fila de los datos devueltos por el SELECT ejecutado sobre la base de datos. Este array puede usar claves numéricas para acceder a cada valor, o el nombre de los campos de la base de datos como claves. En el ejemplo, accedemos al valor cuya clave es "nombre".

La función mysqli_fetch_array puede llevar un segundo parámetro, cuyos valores pueden ser:

- MYSQL_NUM: los datos son devueltos en un array que usa números como claves. Por ejemplo, para acceder a la primera columna de la fila devuelta, la clave será 0.
- MYSQL_ASSOC: los datos son devueltos en un array que usa el nombre de las columnas como claves.
- MYSQL_BOTH: los datos son devueltos en un array que usa tanto números como los nombres de las columnas como claves. Si no se especifica nada, MYSQL_BOTH es el valor por defecto que toma la función mysqli_fetch_array.

Podemos usar la función extract para obtener los valores del array separados, en varias variables cuyo nombre será el mismo que el nombre de las claves del array. Veamos el uso de la función extract con el código anterior:

```
$consulta = "SELECT * FROM Usuario WHERE user='admin' AND
pass='12345'";
$resultado = mysqli_query($con, $consulta)
or die("Error en la ejecución de la consulta");
$fila = mysqli_fetch_array($resultado);
if($fila){
     extract($fila);
     echo "Bienvenido ".$nombre;
}
else{
     echo "El usuario no existe";
}
```

La función mysqli_fetch_array nos devuelve los datos de una única fila en un array. Pero ¿cómo podemos acceder a todas las filas, en caso de que la consulta SELECT devuelva varias? La respuesta es sencilla: usaremos la función mysqli_fetch_array en un bucle. Por ejemplo:

```
$consulta = "SELECT * FROM Alumno WHERE nota>=5";
$resultado = mysqli_query($con, $consulta)
```

```
or die("Error en la ejecución de la consulta");
while($fila = mysqli_fetch_array($resultado)){
     extract($fila);
     echo "Alumno: ".$nombre." Nota:".$nota."<br/>";
}
```

En el ejemplo anterior mostramos el nombre y la nota de todos los alumnos cuya nota sea superior o igual a 5.

Podemos usar igualmente bucles for, pero recordemos que en estos bucles debemos saber el número de iteraciones que haremos. En el caso que estamos tratando, haremos tantas iteraciones como filas tenga el resultado de la operación SELECT. La función mysqli_num_rows nos devuelve el número de filas resultantes de una consulta:

```
$consulta = "SELECT * FROM Alumno WHERE nota>=5";
$resultado = mysqli_query($con, $consulta)
or die("Error en la ejecución de la consulta");
$num_filas = mysqli_num_rows($resultado);
if($num_filas == 0){
     echo "No hay datos a mostrar<br/>";
}
for($i=0;$i<$num_filas;$i++){
     $fila = mysqli_fetch_array($resultado);
     extract($fila);
     echo "Alumno: ".$nombre." Nota:".$nota."<br/>";
}
```

Ejemplo de consulta sobre una base de datos MySQL

2.4. Uso de funciones

En un tema anterior se habló de las funciones y de su utilidad a la hora de evitar repetir código y facilitar la comprensión de nuestro programa. En nuestras aplicaciones vamos a realizar varias veces la consulta de datos a la base de datos, por lo que es una buena práctica usar funciones para estructurar de una forma eficiente nuestro código. Veamos un ejemplo en el que la consulta de datos se realiza en una función:

```
$servidor = "localhost";
$usuario = "root";
$pass = "12345";
$db = "Escuela";
$con = mysqli_connect($servidor, $usuario, $pass, $db)
or die ("Error en la conexión a la BD);
$datos = extraerInformacionAlumno("Daniel", "Santiago");
echo "La nota de este alumno es: ".$datos["nota"];

//Código de la función extraerInformacionAlumno
function extraerInformacionAlumno($nombre, $apellido){
     $consulta = "SELECT * FROM Alumno WHERE
     nombre='$nombre' AND apellido='$apellido'";
     $resultado = mysqli_query($con, $consulta)
     or die("Error en la ejecución de la consulta");
     return mysqli_fetch_array($resultado);
}
```

Con el uso de funciones podremos ejecutar varias veces un mismo grupo de sentencias sin tener que duplicarlas por nuestra aplicación. En el ejemplo estamos buscando la nota del alumno Daniel Santiago, pero en otra parte de nuestra aplicación podremos usar el mismo código para obtener los datos de otro alumno. Bastará con cambiar los parámetros que pasamos en la llamada a la función extraerInformacionAlumno.

3. Insertar y actualizar información en una base de datos

3.1. Introducción

En este apartado veremos cómo insertar los datos que usará nuestra aplicación en una base de datos, para después poder usar esa información con las herramientas que hemos visto en el apartado anterior.

3.2. Preparar los datos para insertarlos en la base de datos

Lo primero que tenemos que hacer para almacenar en la base de datos la información introducida por el usuario en un formulario es asegurarnos que los datos están en el formato

correcto. En nuestra base de datos habremos definido los datos según su tipo: podrán ser alfanuméricos (CHAR, VARCHAR), numéricos (INT, DECIMAL), fechas (DATE, DATETIME)… También podemos añadir restricciones adicionales, como el tamaño máximo de una cadena de caracteres (maxlength).

Por lo general, los datos que introduce el usuario en un formulario se almacenan en la base de datos tal y como están. Por ejemplo, los usuarios digitan sus nombres en el formulario, y el programa los almacena. Sin embargo, en algunos casos, los datos deben cambiarse antes de ser almacenados. Por ejemplo, si un usuario introduce la fecha en un formulario en tres listas de selección separadas para día, mes y año, los valores en los tres campos deben juntarse en una sola variable. Podemos hacerlo de la siguiente forma:

```
$fecha = $_POST["dia"]."-".$_POST["mes"]."-
".$_POST["anyo"];
```

Otro caso en el que podemos querer cambiar los datos antes de almacenarlos es cuando estamos guardando números de teléfono. Los usuarios introducen los números de teléfono en una variedad de formatos, usando paréntesis, guiones, espacios, etc. En vez de almacenar esta variedad de formatos en la base de datos, sólo querremos guardar los números. Veamos el uso de la función ereg_replace para quedarnos con números de teléfono en los que sólo aparezcan números:

```
$telefono = ereg_replace("[]().- ", "",
$_POST["telefono"]);
```

La función ereg_replace usa tres parámetros:

- En el primero pondremos una cadena con todos los caracteres que queremos sustituir. En nuestro ejemplo, corchetes, paréntesis, punto, guión y espacio.
- En el segundo pondremos la cadena por la que queremos sustituir cualquier carácter de la cadena anterior. En nuestro caso, queremos eliminar los caracteres de la primera cadena, por tanto en el segundo parámetros ponemos una cadena vacía (“”).
- El tercer parámetro contiene la cadena sobre la cual vamos a operar. En nuestro caso, el teléfono introducido por el usuario en un formulario.

Otras funciones útiles para depurar los datos que introducen los usuarios en un formulario son:

- strip_tags: esta función retira todo el texto encerrado entre < y >. Busca un < de apertura y lo elimina junto con todo lo demás, hasta encontrar un > de cierre o llegar al final de la cadena. En esta función también podemos incluir etiquetas específicas que desea permitir. Por ejemplo, si queremos borrar todas las etiquetas de una cadena de caracteres excepto <b> <i>:

```
$apellido = strip_tags($apellido, "<b><i>");
```

- htmlspecialchars: esta función cambia algunos caracteres especiales que tienen significado para HTML a un formato HTML que permita mostrarlos son ningún significado especial. Los cambios son:
  - < se convierte en <
  - > se convierte en >

- & se convierte en &

De esta forma, los caracteres < y > se pueden mostrar en una página web sin ser interpretados por HTML como etiquetas. El siguiente ejemplo cambia estos caracteres especiales:

```
$apellido = htmlspecialchars($apellido);
```

- trim: esta función elimina los espacios que hay al inicio o al final de un texto, para que no sean almacenados. Se usa de la siguiente manera:

```
$apellido = trim($_POST["apellido"]);
```

3.3. Insertar información en la base de datos

Para insertar datos en una base de datos se usa la sentencia INSERT. Recordemos su sintaxis:

```
INSERT INTO nombreTabla (columna1, columna2, columna3…)
VALUES (valor1, valor2, valora3…);
```

Para ejecutar esta instrucción SQL con PHP, usaremos la función mysqli_query. Veamos un ejemplo de uso:

```
$nombre = $_POST["nombre"];
$apellido = $_POST["apellido"];
$consulta = "INSERT INTO Usuario (nombre, apellido)
VALUES ('$nombre', '$apellido')";
$resultado = mysqli_query($consulta) or die("No se puede
ejecutar la consulta");
echo "Usuario creado correctamente";
```

El código anterior guarda en las variables $nombre y $apellido los valores que ha introducido el usuario en sendos campos de un formulario. A continuación, construimos la consulta SQL incorporando estos valores, y la guardamos en la variable $consulta. La función mysqli_query ejecutará la consulta. Obviamente, debemos haber conectado con la base de datos previamente, usando las funciones vistas previamente.

3.4. Actualizar información de la base de datos

Para actualizar datos de una base de datos se usa la sentencia UPDATE. Su sintaxis es:

```
UPDATE nombreTabla SET columna1=valor1, columna2=valor2,
columna3=valora3 WHERE condiciones;
```

Es importante recordar que la cláusula WHERE es necesaria en los UPDATE a no ser que queramos modificar todas las filas de la tabla.

El siguiente código ejemplifica el uso de la sentencia UPDATE:

```
$nombre = $_POST["nombre"];
$apellido = $_POST["apellido"];
$telefono = $_POST["telefono"];
```

```
$consulta = "UPDATE Usuario SET telefono = $telefono
WHERE nombre='$nombre' AND apellido='$apellido'";
$resultado = mysqli_query($consulta) or die("No se puede
ejecutar la consulta");
echo "Datos actualizados correctamente";
```

En este código actualizamos el teléfono del usuario con el nombre y apellido indicados en un formulario.

Ejemplo de inserción de datos sobre una base de datos MySQL

3.5. Eliminar información de la base de datos

La sentencia para eliminar datos de la base de datos es DELETE. Su sintaxis es:

```
DELETE FROM nombreTabla WHERE condiciones;
```

Es muy importante no olvidarnos de poner la cláusula WHERE, ya que borraríamos toda la información de la tabla.

Vamos a ver un ejemplo de borrado del usuario cuyo nombre y apellido vienen especificados en un formulario:

```
$nombre = $_POST["nombre"];
$apellido = $_POST["apellido"];
$consulta = "DELETE FROM Usuario WHERE nombre='$nombre'
AND apellido='$apellido'";
$resultado = mysqli_query($consulta) or die("No se puede
ejecutar la consulta");
echo "Usuario borrado";
```

3.6. Funciones útiles para trabajar con MySQL

En la tabla siguiente veremos algunas de las muchas funciones que ofrece PHP para trabajar con los datos de una base de datos MySQL. Como puede verse, las hay de muy diversas funcionalidades. Se recomienda consultar el manual oficial de PHP para descubrir más funciones y ver ejemplos detallados de su uso.

Función	Descripción
mysqli_fetch_array ($result)	Extrae una fila de datos de $result en forma de array, o devuelve null si no tiene filas.
mysqli_num_rows ($result)	Devuelve el número de filas de $result.
mysqli_fetch_row ($result)	Extrae una fila de datos de $result en forma de array enumerado, o devuelve null si no hay más filas.
mysqli_num_fields ($result)	Devuelve el número de campos de $result.
mysqli_affected_rows ($con)	Devuelve el número de filas afectadas en la última operación SQL, o -1 si se produjo algún error.
mysqli_fetch_assoc ($result)	Extrae una fila de datos de $result en forma de array asociativo, o devuelve null si no hay más filas.

Test de autoevaluación

¿Con qué función conectaremos a una base de datos en PHP 5.5 o posteriores y MySQL?

a) mysql_connect
b) mysqli_connect
c) mysql
d) mysql_select_db

¿Cuál de los siguientes parámetros NO es obligatorio especificar en una llamada a la función mysqli_connect?

a) $host
b) $user
c) $password
d) $dbname

¿Con qué función podemos ejecutar sentencias SQL en PHP?

a) mysqli_select
b) mysqli_connect
c) mysqli_query
d) mysqli_execute

¿Qué tipo de dato devuelve la instrucción mysqli_query("INSERT INTO…")?

a) Un booleano
b) Un objeto que contiene el registro insertado en la base de datos
c) No devuelve nada
d) Un entero con el número de registros insertados

¿Qué tipo de dato devuelve la instrucción mysqli_query("SELECT…")?

a) Un booleano
b) Un entero con el número de registros seleccionados
c) No devuelve nada
d) Un ResultSet

Recursos y enlaces

- Página de descarga de MySQL
- Manual oficial PHP: trabajar con MySQL

Conceptos clave

- **Funciones mysqli_**: para versión de PHP a partir de la 5.5.0 (incluida) los nombres de las funciones tienen el prefijo “mysqli_”.
- **mysqli_connect**: función para conectar con un servidor MySQL.
- **mysqli_close:** función para cerrar la conexión con un servidor MySQL.
- **mysqli_query:** función para ejecutar sentencias SQL sobre la base de datos como son SELECT, INSERT, UPDATE o DELETE. El resultado devuelto será: TRUE si se ejecutan exitosamente operaciones SQL que no obtienen datos, como INSERT, UPDATE o DELETE. Si se produce algún error en la operación (debido, por ejemplo, a que la sintaxis es errónea) la función devolverá FALSE. Si ejecutamos operaciones que devuelven datos, como SELECT, se generará un identificador de donde podremos obtener los datos devueltos por la instrucción SELECT. Si la operación falla, obtendremos el valor FALSE.

Ponlo en práctica

Actividad 1

Realiza una aplicación que se conecte a la base de datos Escuela y recorra todos los datos de la tabla Alumno (dni, nombre, apellidos, nota_media) mostrándolos por pantalla.

SOLUCIÓN

Actividad 2

Realiza una aplicación web en la que, mediante un formulario, se pidan los datos para dar de alta a un nuevo alumno en la base de datos Escuela. Haz una página con el formulario y otra con las operaciones en PHP que inserten un nuevo registro en la tabla Alumno con los datos del formulario.

SOLUCIÓN

Actividad 3

Realiza una aplicación web en la que se cree un formulario de forma dinámica a partir de los campos de la tabla Alumno de la base de datos Escuela.

SOLUCIÓN

Actividad 4

Realiza una aplicación web en la que se cree una lista de selección de un formulario cuyas opciones provienen de los datos de la tabla Categoría de la base de datos Tienda.

SOLUCIÓN