Programación Web Día

LUIS EDIN IBARGUEN MOSQUERA

Taller N°. 2

CONSULTAS

ING. CARLOS ALBERTO LONDOÑO LOAIZA

CORPORACIÓN DE ESTUDIOS

TECNOLÓGICOS

DEL NORTE DEL VALLE

PROGRAMA

CARTAGO

2016

1. Crear una base de datos en MYSQL

R/. Para empezar, crearemos una base de datos para nosotros solos, y la llamaremos "prueba". Para crear una base de datos se usa una sentencia [CREATE DATABASE](http://mysql.conclase.net/curso/?sqlsen=CREATE_DATABASE):

mysql> CREATE DATABASE prueba;

Query OK, 1 row affected (0.03 sec)

mysql>

Podemos averiguar cuántas bases de datos existen en nuestro sistema usando la sentencia [SHOW DATABASES](http://mysql.conclase.net/curso/?sqlsen=SHOW_DATABASES):

mysql> SHOW DATABASES;

+--------------------+

| Database |

+--------------------+

| mysql |

| prueba |

| test |

+--------------------+

3 rows in set (0.00 sec)

mysql>

**B).** Hasta ahora era habitual usar la función de PHP *mysql\_connect()* para crear conexiones contra bases de datos MySQL. Esta función fue declarada obsoleta en de PHP 5.5.0 y **a partir de PHP 7.0.0 será eliminada**.

Por este motivo, es conveniente que empecemos a usar la nueva metodología orientada a objetos. Las nuevas funciones tienen nombres parecidos y el funcionamiento es muy similar. Para crear una conexión a una base de datos deberemos hacerlo con la clase *mysqli*:

/\*

\* Datos de conexión a MySQL

\*/

$db\_database = 'nombre\_bd';

$db\_hostname = 'localhost';

$db\_username = 'usuario\_bd';

$db\_password = 'P@ssw0rd';

/\*

\* Creación del objeto mysqli

\*/

$mysqli = new mysqli($db\_hostname, $db\_username, $db\_password, $db\_database);

/\*

\* Buscamos posibles errores en la conexión

\*/

if (mysqli\_connect\_errno()) {

printf("Falló la conexión: %s\n", mysqli\_connect\_error());

exit();

}

Si no hay ningún error, el objeto *$mysqli* será el que nos servirá para lanzar consultas a la base de datos. Por ejemplo, si queremos obtener datos y mostrarlos por pantalla, en primer lugar necesitaremos crear la consulta y ejecutarla con el método *query* del objeto *$mysqli*:

/\*

\* Creación de la consulta SQL

\*/

$query = "SELECT \* FROM productos";

/\*

\* Guardamos el resultado que devuelve MySQL

\*/

$resultado = $mysqli->query($query)

Una vez guardado el resultado de la consulta, tenemos varias formas de iterar sobre el mismo, por ejemplo a través de un array asociativo, que conseguimos con el método *fetch\_assoc()* del objeto*$resultado*:

/\*

\* Iteramos sobre el resultado y mostramos los datos

\*/

while ($producto = $resultado->fetch\_assoc()) {

echo $producto["campo1"];

echo $producto["campo2"];

echo $producto["campo3"];

}

/\*

\* Liberamos los recursos reservados

\*/

$resultado->free();

$mysqli->close();

Es una buena práctica a nivel de rendimiento **liberar el conjunto de resultados y cerrar la conexión una vez terminemos**. Como has visto, esta metodología es muy parecida a la obsoleta, por lo que no supone demasiado tiempo aprender a usarla. Además, para iniciados en PHP la curva de aprendizaje es menor.

Si tienes más dudas sobre cómo usar estos métodos de PHP u otros, no dudes en dejarnos un comentario.

**C).** **EJEMPLO DE INSERCIÓN DE DATOS EN UNA TABLA**

Escribe este código en un editor y guárdalo en un archivo con un nombre como ejemplo1.php. Luego ejecuta el código (una sola vez, ten en cuenta que si lo ejecutas varias veces insertarás varias veces repetidamente los datos en la base de datos). Recuerda también que debes cambiar y poner tus propios valores en las funciones mysqli\_connect y mysqli\_select\_db.

|  |
| --- |
| <?php //Ejemplo curso PHP aprenderaprogramar.com  $link = mysqli\_connect("sql203.byethost7.com", "b7\_10356956", "\*\*\*\*\*");  mysqli\_select\_db($link, "b7\_10356956\_mibasededatos");  $tildes = $link->query("SET NAMES 'utf8'"); //Para que se inserten las tildes correctamente  mysqli\_query($link, "INSERT INTO agenda VALUES ('Pedro2', 'Gómez Gómez', 'C/ Buenaventura 54', '699887766', 35, 1.98)");  mysqli\_query($link, "INSERT INTO agenda VALUES ('Antonio2', 'Delgado Folken', 'C/ El Milagroso 77', '699667744', 88, 1.71)");  mysqli\_close($link); // Cerramos la conexion con la base de datos  echo 'Los datos han sido insertados en la base de datos';  ?> |

Nota: si usáramos sintaxis no mysqli se usaría mysql\_query ("INSERT INTO nombreDeLaTabla VALUES (datoCol1, datoCol2, … datoColN ", $link).

**D).**

Para mostrar cómo se actualiza un registro presente en nuestra base de datos, vamos a hacerlo a partir de un caso un poco más complejo para que empecemos a familiarizarnos con estas operaciones. Realizaremos un par de scripts que permitan cambiar el numero de teléfono de las distintas personas presentes en nuestra base. El nombre de estas personas, así como el nuevo numero de teléfono, serán recogidos por medio de un formulario.

El archivo del formulario va a ser esta vez un script PHP en el que efectuaremos una llamada a nuestra base de datos para construir un menú desplegable donde aparezcan todos los nombres. La cosa quedaría así:

<HTML>  
<HEAD>  
<TITLE>Actualizar1.php</TITLE>  
</HEAD>  
<BODY>  
<div align="center">  
<h1>Actualizar un registro</h1>  
<br>  
<?  
//Conexion con la base  
mysql\_connect("localhost","tu\_user","tu\_password");  
  
//selección de la base de datos con la que vamos a trabajar   
mysql\_select\_db("mi\_base\_datos");   
  
echo '<FORM METHOD="POST" ACTION="actualizar2.php">Nombre<br>';  
  
//Creamos la sentencia SQL y la ejecutamos  
$sSQL="Select nombre From clientes Order By nombre";  
$result=mysql\_query($sSQL);  
  
echo '<select name="nombre">';  
  
//Generamos el menu desplegable  
while ($row=mysql\_fetch\_array($result))  
{echo '<option>'.$row["nombre"];}  
?>  
</select>  
<br>  
Teléfono<br>  
<INPUT TYPE="TEXT" NAME="telefono"><br>  
<INPUT TYPE="SUBMIT" value="Actualizar">  
</FORM>  
</div>  
  
</BODY>  
</HTML>

La manera de operar para construir el menú desplegable es la misma que para visualizar la tabla. De nuevo empleamos un bucle *while* en combinación con la función*mysql\_fetch\_array* lo que nos permite mostrar cada una de las opciones.

El script de actualización será muy parecido al de inserción:

**E).** **SENTENCIA DELETE Y CLÁUSULA WHERE**

La sentencia DELETE se suele usar para borrar unos registros de una tabla que cumplen una o varias condiones. Vamos a utilizar una sintaxis de este tipo:

|  |
| --- |
| mysqli\_query($link, " DELETE FROM nombreTabla WHERE columna (>, <, =, etc.) valorEspecificado "); |

|  |
| --- |
| <?php //Ejemplo curso PHP aprenderaprogramar.com  $link = mysqli\_connect("sql203.byethost7.com", "b7\_10356956", "\*\*\*\*\*");  mysqli\_select\_db($link, "b7\_10356956\_mibasededatos");  mysqli\_query($link, "DELETE FROM agenda2 WHERE altura > 1.90");  mysqli\_close($link); // Cerramos la conexion con la base de datos  echo 'Se ha ejecutado la consulta sobre la base de datos';  ?> |

F). **Consulta de la BD**

<html>

<body>

<?php

$link = mysql\_connect("localhost", "nobody");

mysql\_select\_db("mydb", $link);

$result = mysql\_query("SELECT nombre, email FROM agenda", $link);

echo "<table border = '1'> \n";

echo "<tr><td>Nombre</td><td>E-Mail</td></tr> \n";

while ($row = mysql\_fetch\_row($result)){

echo ""<tr><td>$row[0]</td><td>$row[1]</td></tr> \n";

}

echo "</table> \n";

?>

</body>

</html>

En este **script** hemos introducido dos novedades, la más obvia es la sentencia de control while(), que tiene un funcionamiento similar al de otros lenguajes, ejecuta una cosa mientras la condición sea verdadera. En esta ocasión while() evalua la función mysql\_fetch\_row(), que devuelve un array con el contenido del registro actual (que se almacena en $row) y avanza una posición en la lista de registros devueltos en la consulta **SQL**.

La función mysql\_fetch\_row() tiene un pequeño problema, es que el **array** que devuelve sólo admite referencias numéricas a los campos obtenidos de la consulta. El primer **campo** referenciado es el **0**, el segundo el **1** y así sucesivamente. En el siguiente **script** solucionaremos este pequeño inconveniente.

**G).** **REQUIRE**

require es idéntico a include excepto que en caso de fallo producirá un error fatal de nivel E\_COMPILE\_ERROR. En otras palabras, éste detiene el script mientras que include sólo emitirá una advertencia (E\_WARNING) lo cual permite continuar el script.

**H)**. **REQUIRE\_ONCE**

La sentencia require\_once es idéntica a require excepto que PHP verificará si el archivo ya ha sido incluido y si es así, no se incluye (require) de nuevo.

Ver la documentación de include\_once para información sobre el comportamiento de \_once, y como difiere de sus hermanos no \_once.

**I)**. La sentencia **include** incluye y evalúa el archivo especificado.

La siguiente documentación también se aplica a require.

Los archivos son incluidos con base en la ruta de acceso dada o, si ninguna es dada, el include\_path especificado. Si el archivo no se encuentra en el include\_path, include finalmente verificará en el propio directorio del script que hace el llamado y en el directorio de trabajo actual, antes de fallar. El constructor include emitirá una advertencia si no puede encontrar un archivo, éste es un comportamiento diferente al de require, el cual emitirá un error fatal.

**J).** La sentencia **include\_once** incluye y evalúa el fichero especificado durante la ejecución del script. Tiene un comportamiento similar al de la sentencia include, siendo la única diferencia de que si el código del fichero ya ha sido incluido, no se volverá a incluir, e include\_once devolverá TRUE. Como su nombre indica, el fichero será incluido solamente una vez.

include\_once se puede utilizar en casos donde el mismo fichero podría ser incluido y evaluado más de una vez durante una ejecución particular de un script, así que en este caso, puede ser de ayuda para evitar problemas como la redefinición de funciones, reasignación de valores de variables, etc.