Tutorial de PHP y MYSQL Desde Cero

www.clubdeprogramacion.com

TUTORIAL DE PHP Y MYSQL

Desde Cero

Exclusión de Responsabilidad

Este reporte ha sido escrito para proveer información sobre INGRESA EL TOPICO ACA. Se han hecho todos los esfuerzos posibles para que este reporte sea lo más exacto y adecuado posible. Sin embargo puede que existan errores ortográficos y/o de contenido. La información está actualizada a la fecha de la publicación y está basada en la experiencia/investigación del autor. Por esta razón se debe considerar como una guía únicamente.

El propósito del reporte es educativo y tanto el autor como la editorial no garantizan que el contenido sea 100% exacto por lo que no se responsabilizan por ningún tipo de error u omisión.

Tanto el autor como la editorial se liberan de cualquier tipo de responsabilidad a cualquier persona o entidad por daños o pérdidas causados o supuestamente causados, directa o indirectamente por este reporte.

Si no está de acuerdo con le exclusión anterior por favor no lea y devuelva este reporte para obtener un reembolso completo.

Aclaración de Conexión y Divulgación

Usted debe asumir que el autor y el editor tienen una relación de afiliados y/o algún otro tipo de conexión material con los proveedores de productos y servicios mencionados en esta guía y pueden ser compensados cuando usted compre de un proveedor. Le recomendamos informase adecuadamente sobre cualquier producto o servicio que vaya a comprar en línea o fuera de ella.

Mensaje del Autor

Espero que este tutorial de php y mysql pueda servirte a entender el mundo de la programación web con este maravilloso lenguaje, cada ejercicio está pensado de forma sencilla y práctica. Espero realmente que pueda ser de gran utilidad en tu vida profesional.

Atte.
César Mayta
Autor del tutorial

Tabla de Contenido

INTRODUCCION A PHP

¿QUE ES PHP?

PHP es un lenguaje de script del lado del servidor.
Los scripts PHP están incrustados en los documentos HTML y el servidor los interpreta y ejecuta antes de servir las páginas al cliente
El cliente no ve el código PHP sino los resultados que produce

¿Cómo funciona PHP?

A diferencia de Java o JavaScript que se ejecutan en el navegador, PHP se ejecuta en el servidor, por eso nos permite acceder a los recursos que tenga el servidor como por ejemplo podría ser una base de datos. El programa PHP es ejecutado en el servidor y el resultado enviado al navegador. El resultado es normalmente una página HTML.

MI PRIMER SCRIPT

Vamos a crear un script sencillo para entender como funciona php

```
<HTML>
<HEAD>
<TITLE>Mi primer programa en PHP</TITLE>
</HEAD>
<BODY>
<?php echo (“<P>Hola mundo</P>”); ?>
</BODY>
</HTML>
```

Parece una pagina html comun pero lo que se encuentra dentro de los símbolos <?php y ?> es codigo php que el servidor interpretara para mostrar las etiquetas correspondiente

CONOCIENDO LA SINTAXIS DE PHP

Ahora aprenderemos rápidamente la sintaxis de este lenguaje, es decir cómo se compone su lenguaje.

VARIABLES:

Las variables son espacios de memoria que sirven para almacenar datos que pueden variar durante el progreso del programa, en PHP deben estar precedidas por signo dólar ($), y le asignamos contenido con el signo igual (=).
PHP distingue entre mayúsculas y minúsculas, por lo que no es lo mismo $myvar que $Myvar, éstas son dos variables totalmente distintas.

```
<html>
<body>
<?php
 $myvar = "SEVILLA";
 $Myvar = "MADRID";
 //Esto imprimirá SEVILLA
 echo($myvar.”<br>”);
 //Esto imprimirá MADRID
 echo($Myvar.”<br>”);
?>
</body>
</html>
```

Como ven se ha utilizado dos formas de escribir echo, en mayúsculas y en minúsculas, para indicar que PHP no las distingue a la hora de usar funciones o sentencias del lenguaje.

CONSTANTES:

Las constantes son similares a las variables, con la salvedad de que no llevan el signo dólar delante, y sólo la podemos asignar una vez. Para definir una constantes usaremos la función define como sigue:

```
<html>
<body>
<?php
define ("CONSTANTE", "Hola Mundo");
printf (CONSTANTE);
```



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

OPERADORES:

ARITMÉTICOS:

$a + $b Suma
$a - $b Resta
$a * $b Multiplicación
$a / $b &ss=codigoenlinea>$a / $b División
$a % $b Resto de la división de $a por $b
$a++ Incrementa en 1 a $a
$a-- Resta 1 a $a

CONCATENADORES:

La concatenación de cadenas se maneja con el punto.

```
$a = "Hola";
$b = $a . "Mundo"; // Ahora $b contiene "Hola Mundo"
```

En este punto hay que hacer una distinción, la interpretación que hace PHP de las simples y dobles comillas. En el segundo caso PHP interpretará el contenido de la cadena.

```
$a = "Mundo";
echo = 'Hola $a'; //Esto escribirá "Hola $a"
echo = "Hola $a"; //Esto escribirá "Hola Mundo&q; //Esto escribirá "Hola Mundo"
```

COMPARADORES:

$a < $b $a menor que $b
$a > $b $a mayor que $b
$a <= $b $a menor o igual que $b
$a >= $b $a mayor o igual que $b
$a == $b $a igual que $b
$a != $b $a distinto que $b

LOGICOS:

$a AND $b Verdadero si ambos son verdadero
$a && $b Verdadero si ambos son verdadero
$a OR $b Verdadero si alguno de los dos es verdadero

$a !! $b Verdadero si alguno de los dos es verdadero
$a XOR $b Verdadero si sólo uno de los dos es verdadero<BP; de verdadero
!$a Verdadero si $a es falso, y recíprocamente

DE ASIGNACION:

$a = $b Asigna a $a el contenido de $b
$a += $b Le suma a $b a $a
$a -= $b Le resta a $b a $a
$a *= $b Multiplica $a por $b y lo asigna a $a
$a /= $b Divide $a por $b y lo asigna a $a
$a .= $b Añade la cadena $b a la cadena $a

SENTENCIAS CONDICIONALES:

Las sentencias de control permiten ejecutar bloque de códigos dependiendo de unas condiciones. Para PHP el 0 es equivalente a Falso y cualquier otro número es Verdadero.

IF…ELSE

La sentencia IF…ELSE permite ejecutar un bloque de instrucciones si la condición es Verdadera y otro bloque de instrucciones si ésta es Falsa. Es importante tener en cuenta q instrucciones si ésta es Falsa. Es importante tener en cuenta que la condición que evaluemos ha de estar encerrada entre paréntesis (esto es aplicable a todas las sentencias de control).

```
if (condición) {
   Este bloque se ejecuta si la condición es VERDADERA
} else {
   Este bloque se ejecuta si la condición es FALSA
}
```

Existe una forma sencilla de usar la sentencia IF cuando no tenemos que usar el ELSE y solo tenemos que ejecutar una línea de código.
if ($a > 4) echo "$a es mayor que 4";

IF…ELSEIF…ELSE

La sentencia IF…ELSEIF…ELSE permite ejecuta varias condiciones en cascada. Para este caso veremos un ejemplo, en el que utilizaremos los

operadores lógicos.

```
<?php
if ($nombre == ""){ echo "Tú no tienes nombre";
} elseif (($nombre=="eva") OR ($nombre=="Eva")) {
   echo " echo "Tu nombre es EVA";<
} else {
   echo "Tu nombre es " . $nombre;
}
```

SWITCH…CASE…DEFAULT

Una alternativa a IF…ELSEIF…ELSE, es la sentencia SWITCH, la cuál evalúa y compara cada expresión de la sentencia CASE con la expresión que evaluamos, si llegamos al final de la lista de CASE y encuentra una condición Verdadera , ejecuta el código de bloque que haya en DEFAULT. Si encontramos una condición verdadera debemos ejecutar un BREAK para que la sentencia SWITCH no siga buscando en la lista de CASE. Veamos un ejemplo.

```
<?php
switch ($dia) {
case "Lunes":

echo "Hoy es Lunes";
break;
case "Martes":
echo "Hoy es Martes";
break;
case "Miercoles":
echo "Hoy es Miercoles";
break;
case "Jueves":
echo "Hoy es Jueves";
break;
case "Viernes":
echo "Hoy es Viernes";
break;
case "Sábado":
echo "Hoy es Sábado";
break;
case "Domingo":
echo "Hoy es Domingo";
```



```
break;
default:
echo "Esa cadena no corresponde a ningún día de la semana";
}
?>
```

SENTENCIAS CICLICAS O BUCLES:

nos permiten iterar conjuntos de instrucciones, es decir repetir la ejecución de un conjunto de instrucciones mientras se cumpla una condición.

WHILE:

La sentencia WHILE ejecuta un bloque de código mientras se cumpla una determinada condición.

```
<?php
$num = 1;
while ($num < 5) {
echo $num;
$num++
}
?>
```

Podemos romper un bucle WHILE utilizando la sentencia BREAK.

```
<?php
$num = 1;
while ($num < 5) {
echo $num;
if ($num == 3){
echo "Aquí nos salimos \n";
break
}
$num++

}
?>
```

DO...WHILE:

Esta sentencia es similar a WHILE, salvo que con esta sentencia primero ejecutamos el bloque de código y después se evalúa la condición, por lo que el bloque de código se ejecuta siempre al menos una vez.

```
<?php
```

```
$num = 1;
do {
echo $num;
if ($num == 3){
echo "Aquí nos salimos \n";
break
}
$num++
} while ($num < 5);
?>
```

FOR:

El bucle FOR no es estrictamente necesario, cualquier bucle FOR puede ser sustituido fácilmente por otro WHILE. Sin embargo, el bucle FOR resulta muy útil cuando debemos ejecutar un bloque de código a condición de que una variable se encuentre entre un valor mínimo y otro máximo. El bucle FOR también se puede romper mediante la sentencia BREAK.

```
<?php
for ($num = 1; Snum <=5; $num++){
echo $num;
if ($num == 3){
echo "Aquí nos salimos \n";
break
}
}
?>
```

ARRAYS:

Las tablas (o array en inglés), son muy importantes en PHP, ya que generalmente, las funciones que devuelven varios valores, como las funciones ligadas a las bases de datos, lo hacen en forma de tabla.
En PHP disponemos de dos tipos de tablas. El primero sería el clásico, utilizando índices:

```
<?php
$ciudad[] = "París";
$ciudad[] = "París";
$ciudad[] = "Roma";
$ciudad[] = "Sevilla";
$ciudad[] = "Londres";
print ("yo vivo en " . $ciudad[2] . "<BR>\n");
```

```
?>
```

Esta es una forma de asignar elementos a una tabla, pero una forma más formal es utilizando la función array

```
$ciudad = array("París", "Roma", "Sevilla", "Londres");
//contamos el número de elementos de la tabla
$numelentos = count($ciudad);
//imprimimos todos los elementos de la tabla
for ($i=0; $i < $numelentos; $i++)
{
print ("La ciudad $i es $ciudad[$i] <BR>\n");
}
?>
```

Si no se especifica, el primer índice es el cero, pero podemos utilizar el operador => para especificar el indice inicial.

```
$ciudad = array(1=>"París", "Roma", "Sevilla", "Londres");
```

Un segundo tipo, son las tablas asociativas, en las cuáles a cada elemento se le asigna un valor (key) para acceder a él.

Para entenderlo, que mejor que un ejemplo, supongamos que tenemos una tabla en al que cada elemento almacena el número de visitas a nuestra web por cada día de la semana.

Utilizando el método clásico de índices, cada día de la semana se representaría por un entero, 0 para lunes, 1 para martes, etc.

```
$visitas[0] = 200;
$visitas[1] = 186;
```

si usamos las tablas asociativas sería

```
$visitas["lunes"] = 200;
$visitas["martes"] = 186;
```

o bien,

```
$visitas = array("luodigo">$visitas = array("lunes"=>200; "martes"=>186);
```

Ahora bien, recorrer una tabla y mostrar su contenido es sencillo utilizando los índices, pero ¿cómo hacerlo en las tablas asociativas?. La manipulación de las tabas asociativas se a hace través de funciones que actúan sobre un puntero interno que indica la posición. Por defecto, el puntero se sitúa en el primer elemento añadido en la tabla, hasta que es movido por una función:

current - devuelve el valor del elemento que indica el puntero
pos - realiza la misma función que current
reset - mueve el puntero al primer elemento de la tabla
end - mueve el puntero al último elemento de la tabla
next - mueve el puntero al elemento siquiente
prev - mueve el puntero al elemento anterior
count&n count - devuelve el número de elementos de una tabla.

Veamos un ejemplo de las funciones anteriores:

```
<?php
$semana = array("lunes", "martes", "miércoles", "jueves", "viernes",
"sábado", "domindo");
echo count($semana); //7
//situamos el puntero en el primer elemento
reset($semana);
echo current($semana); //lunes
next($semana);
echo pos($semana); //martes
end($semana)
echo pos($semana); //domingo
prev($semana);
echo current($semana); //sábado
?>
```

Recorrer una tabla con las funciones anteriores se hace un poco lioso, para ello se recomienda utilizar la función each().

```
<?php
$visitas = array("lunes"=>200, "martes"=>186, "miércoles"=>190,
"jueves"=>175);
reset($visitas);
while (list($clave, $valor) = each($visitas))
{
echo "el día $clave ha tenido $valor visitas<BR>";
}
?>
```

La función each()devuelve el valor del elemento actual, en este caso, el valor del elemento actual y su clave, y desplaza el puntero al siguiente, cuando llega al final devuelve FALSO, y termina el bucle while().

FUNCIONES:

Muchas veces, cuando trabajamos en el desarrollo de una aplicación, nos surge la necesidad de ejecutar un mismo bloque de código en diferentes partes de nuestra aplicación. Una Función no es más que un bloque de código al que le pasamos una serie de parámetros y nos devuelve un valor. Como todos los lenguaje de programación, PHP trae una gran cantidad de funciones para nuestro uso, pero las funciones más gran cantidad de funciones para nuestro uso, pero las funciones más importantes son las que nosotros creamos.
Para declara una función debemos utilizar la instrucción "function" seguido del nombre que le vamos a dar, y después entre paréntesis la lista de argumentos separados por comas, aunque también habrá funciones que no recojan ningún argumento.

```
function nombre_de_funcion (arg_1, arg_2, ..., arg_n)
{
bloque de código
}
```

Cualquier instrucción válida de PHP puede aparecer en el cuerpo (lo que antes hemos llamado bloque de código) de una función, incluso otras funciones y definiciones de clases.
En PHP no podemos redefinir una función previamente declarada, y además en PHP3, las funciones deben definirse siempre antes de que se invoquen, en PHP4 este requerimiento ya no existe.

LA INSTRUCCIÓN RETURN:

Cuando invocamos una función, la ejecución del programa pasa a ejecutar las líneas de código que contenga la función, y una vez terminado, el programa continua su ejecución desde el punto en que fué llamada la función.
Existe una manera de terminar la ejecución de la función aunque aún haya código por ejecutar, mediante el u haya código por ejecutar, mediante el uso de la instrucción return terminamos la ejecución del código de una función y devolvemos un valor. Podemos tener varios return en nuestra función, pero por lo general, cuantos más return tengamos menos reutilizable será nuestra función.

```
<?php
function mayor ($x, $y)
{
 if ($x > $y) {
return $x." es mayor que ".$y;
 } else {
return $y." es mayor que ".$x;
 }
}
?>
```

Aunque quedariá mejor:

```
<?php
function mayor ($x, $y)
{
$msg = "";
if ($x > $y) {
$msg = $x." es mayor que ".$y;
} else {
$msg = $y." es mayor que ".$x;
}
return $msg;
}
?>
```

Con la instrución return puede devolverse cualquier tipo de valor, incluyendo tablas y objetos. PHP solo permite a las funciones devolver un valor, y para solventar este pequeño problema, si queremos que nuestra función devuelva varios tenemos que utilizar una tabla (array).

PARÁMETROS:

Existen dos formas de pasar los parámetros a una función, por valor o por referencia.
Cuando pasamos una variable por valor a una función, ocurra lo que ocurra en ésta en nada modificará el contenido de la variable. Mientras que si lo hacemos por referencia, cualquier cambio acontecido en la función sobre la variable lo hará para siempre.
E variable lo hará para siempre.
En PHP, por defecto, las variables se pasan por valor. Para hacerlo por referencia debemos anteponer un ampersand (&) a la variable.

```
<?php
function suma ($x, $y)
{
$x = $x + 1;
return $x+$y;
}
$a = 1;
$b = 2;
//parámetros por valor
echo suma ($a, $b); // imprimirá 4
echo $a; // imprimirá 1
//parámetros por referencia
echo suma (&$a, $b); // imprimirá 4
echo $a; //imprimirá 2
?>
```

Si queremos que un parámetro de una función se pase siempre por referencia debemos anteponer un ampersand (&) al nombre del parámetro en la definición de la función.
En PHP podemos definir valores por defecto para los parámetro de una función. Estos valores tienen que ser una expresión constante, y no una variable o miembro de una clase. Además cuando usamos parámetros por defectos, éstos deben estar a la derecha de cualquier parámetro sin valor por defecto, de otra forma PHP nos devolverá un error.

```
<?php
function suma ($x=1, $y)
{
$x = $x + 1;
return $x+$y;
}
?>
```

Si ejecutarmos esta función nos daría error, ya que hemos dado a $x el valor 1 por defecto y la hemos colocado a la izquierda de un parámetro que no tiene valor por defecto. La forma correcta es:

```
<?php
function suma ($y, $x=1)
{
$x = $x + 1;
return $x+$y;
}
?>
```

Cabe destacar que PHP3 no soporta un número variables de parámetros, pero PHP4 sí.
Llegados a este punto, damos un paso atrás y volvemos a las variables, para distinguir entre variables estáticas (static) y globales (global). Las variables estáticas se definen dentro de una función, la primera vez que es llamada dicha función la variable se inicializa, guardando su valor para posteriores llamadas.

```
<?php
function contador ()
{
static $count = 0;
$count = $count + 1;
return $count;
}
echo contador()."<BR>"; // imprimirá 1
echo contador()."<BR>"; // imprimirá 2
echo contador()."<BR>"; // imprimirá 3
?>
```

Las variables globales, no se pueden declarar dentro de una función, lo que hacemos el llamar a una variable que ya ha sido declarada, tomando el valor que tenga en ese momento, pudiendo ser modificado en la función.

```
<?php
var $a = 1;
function ver_a()
{
global $a;
echo $a."<BR>"; // imprimirá el valor de $a
$a += 1; // sumamos 1 a $a
}
echo ver_a(); // imprimirá 1
echo ver_a(); // imprimirá 2
$a = 7;
echo ver_a(); // imprimirá 7
echo ver_a(); // imprimirá 8
?>
```

BIBLIOTECA DE FUNCIONES:

Existen muchas bibliotecas de funciones en PHP
Algunos ejemplos:

- Funciones de manipulación de cadenas
- Funciones de fecha y hora
- Funciones de arrays
- Funciones de ficheros
- Funciones matemáticas
- Funciones de bases de datos
- Funciones de red

Algunas bibliotecas requieren la instalación de componentes adicionales
Todas las funciones de biblioteca están comentadas en la documentación de PHP

FUNCIONES DE MANIPULACIÓN DE CADENAS:

explode()
Divide una cadena en subcadenas

array explode (string separator, string string [, int limit])
rtrim(), ltrim(), trim()
Eliminan caracteres a la derecha, a la izquierda o por ambos lados de una cadena
string rtrim (string str [, string charlist])

strstr()
Busca la primera ocurrencia de una subcadena

strtolower() / strtoupper()
Convierte una cadena a minúscula / mayúscula

strcmp() / strcasecmp()
Compara dos cadenas con/sin distinción de mayúsculas

strlen()
Calcula la longitud de una cadena

FUNCIONES DE FECHA Y HORA:

date()
Formatea una fecha según un formato dado
Ejemplo:

```
$fecha  = date ("j/n/Y H:i");
print ("$fecha");
Resultado:
    26/9/2005 17:36
```

strtotime()
Convierte una fecha en un timestamp de UNIX
Ejemplo:

```
$fecha = date ("j/n/Y", strtotime("5 april 2001"));
print ("$fecha");
Resultado:
    5/4/2001
```

FUNCIONES DE ARRAYS:

array_count_values()
Calcula la frecuencia de cada uno de los elementos de un array

array_search()
Busca un elemento en un array

count()
Cuenta los elementos de un array

sort(), rsort()
Ordena y reindexa un array (r=decreciente)

ksort(), krsort()
Ordena por claves un array (r=decreciente)

AHORRANDO CODIGO:

Cómo ahorramos líneas de código
Por lo general, todos nuestros script tienen partes de código iguales, las funciónes include() y require() nos van ahorrar muchas de estas lineas de código. Ambas funciones hacen una llamada a un determinado fichero pero de dos maneras diferentes, con include(), insertamos lo que contenga el fichero que llamemos de manera literal en nuestro script, mientras que con require(), le decimos que el script necesitará parte de código de se

encuentra en el fichero que llama require().
Como todo esto es un poco lioso, veamos unos ejemplos que nos lo aclará.

```
<?php
include ("header.inc");
echo "Hola Mundo";
include ("footer.inc");
?>
```

Si tenemos encuenta que el fichero header.inc contiene:

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

y el fichero footer.inc contiene:

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

Nuestro script sería equivalente a:

```
<html>
<body>
<?php
<?php
echo "Hola Mundo";
?>
</body>
</html>
```

Ahora veamos el script de ejemplo para la función require():

```
<?php
require ("config.inc");
include ("header.inc");
echo $cadena;
include ("footer.inc");
?>
```

Donde el fichero config.inc tendría algo como ésto:

```
<?php
$cadena = "Hola Mundo";
?>
```

BASE DE DATOS – MYSQL

QUE ES UNA BASE DE DATOS:

Una base de datos (sea cual sea) es un soporte digital que tiene como fin el almacenamiento masivo de información en formato texto plano. Aunque existen formas de guardar otro tipo de datos como imágenes y archivos no es recomendable hacerlo, lo mejor en esos casos es guardar dichos archivos en un directorio y guardar la ruta en la base de datos.

Las bases de datos, son utilizadas en sistemas que requieren una interacción fluida con la aplicación; estas se encargan muchas veces de administrar, editar, y dar de alta. Usualmente la base de datos, está ligada a la programación directa del site, causando que una edición en ella cause una modificación directa en lo que ve el usuario.

Ejemplos de aplicación de una base de datos (entiéndase que están ligadas a un lenguaje dinámico como PHP o ASP):

E – comerce, Agendas, Libros de visitas, foros, portales, etc

ESTRUCTURA NORMAL DE UNA DB:

Una base de datos, a fin de ordenar la información de manera lógica, posee un orden que debe ser cumplido para acceder la información de manera coherente.
Cada base de datos tiene una o más tablas, las cuales cumplen la función de contener los campos.

Un ejemplo de tabla sería “contactos”. Para entender mejor esto, sería como un libro en el excel. Mientras que los campos serían las columnas del excel donde se ordena cada datos insertado al libro. Ejemplo “id, nombres, apellidos, teléfono”. Y luego finalmente tenemos las filas (row), que son la información propiamente dicha.

Por consiguiente una base de datos posee el siguiente orden jerárquico:

- Tablas
- Campos
- Registros

MYSQL:

Es una base de datos con licencia GPL basada en un servidor, puede ser sólo creada por código. Usualmente se utiliza el programa phpMyAdmin como soporte para administrar la base de datos en el nivel de programación (a un usuario normal le resultaría complicado utilizarla desde linea de comandos).

ASPECTOS BÁSICOS DE PHPMYADMIN

phpMyAdmin es una herramienta escrita en PHP diseñada para la administración de las bases de datos MySQL en la Web. Actualmente, phpMyAdmin puede crear y eliminar bases de datos, crear/eliminar/modificar tablas, eliminar/modificar/incluir registros, ejecutar sentencias de SQL, y manejar campos con llaves.

Al acceder a la base de datos deberas logearte con un usuario y password (recuerda el usuario por default y administrador es root y el password es el que colocaste en la instalacion de appserv)

CREAR UNA TABLA EN BASE DE DATOS

La parte izquierda se usa para navegación en phpMyAdmin. Allí podrás encontrar tus bases de datos, en caso tengas alguna. Tan pronto como crees tablas, se mostrarán debajo exactamente de la base de datos a la que pertenecen.

Crearemos una tabla que se llame "Distrito". Usa la opción de Create new table. Escribe en el lugar correcto el nombre de la table: Distrito, y la cantidad de columnas que tendrá la tabla (3 en este caso) en el cajón de texto FIELDS.
Puedes eliminar la tabla usando la función DROP.

Da clic en CONTINUAR y debes ver algo parecido a la gráfica siguiente:

El nombre de cada columna escríbelo en el FIELD.

Ahora captura la información adecuada en los respectivos cajones de texto:

ID int 6
delegacion char 100
colonia char 100

La columna LENGHT indica el valor máximo que se permite en ese campo para captura. Hay distintos tipos de valores que se puede utilizar para la columna TYPE. char, varchar, int, numeric, etc. En este caso utilizaremos char, que tiene un valor fijo y acepta caracteres alfanúmericos. Los tipos TYPES especificados en este ejemplo no necesariamente son los más eficientes a la hora de su uso. Debemos determinar cuidadosamente el tipo de valor que se utilizará dependiendo de la información que se almacenará. La columna ID se usará como llave primaria, o Primary Key, para esta tabla, y se colocará en el tip auto_increment, el cual generará de manera automática un número consecutivo. Coloca el Default en cero 0.

Cuando finalices de capturar los datos, da clic en SAVE, o guardar. Aparecerá una ventana similar a la siguiente:

!Felicidades¡ Has creado tu primera tabla. Los datos correspondientes a los comandos utilizados para crear los campos también se muestra en la pantalla. No es obligatorio ni indispensable, pero con el tiempo empezarás a reconocer comandos de MySQL en los mensajes de phpMyAdmin.

Puedes utilizar DROP para eliminar una tabla o alguno de sus campos.

INSERTANDO DATOS EN UNA TABLA
Da clic en INSERT y una pantalla como la siguiente aparecerá:

Ahora captura los datos correspondientes a cada record. La columna ID es automática de manera que no debes de colocarla.

Es importante saber que si te sientes perdido en phpMyAdmin, es suficiente darle clic a HOME en la parte izquierda y te llevará exactamente al lugar de inicio.

Dale clic a SAVE y la información se almacenará en la tabla Distrito. La imagen que se mostró con anterioridad es la que aparece cuando insertas información en una tabla usando el comando INSERT de SQL. Puedes

añadir más más de un registro por medio de utilizar el botón de radio Insert Another new row, en la forma de inserción.

Cuando finalices la inserción de datos, puedes confirmar su inserción y navegar la información dándole clic a BROWSE. Puedes darle clic a cada registro para editarlo o eliminarlo.

LENGUAJE SQL:

Este es el lenguaje que se utiliza para conectarse a una base de datos. Son sentencias, que realizan un query (consulta) a la DB a fin de que esta les responda con una cantidad de datos limitada según lo buscado. Básicamente, existen muchísimas funciones de SQL, pero detallaré las más usuales, con las cuales se pueden lograr una interacción buena con la DB.

INSERTAR DATOS A UNA TABLA ESPECIFICA:

```
INSERT
INTO `Nombre Tabla` (` Nombre Campo ` , ` Nombre Campo ` , `Nombre
Campo`) VALUES ('Valor', ' Valor', ' Valor ');
```

EDITAR DATOS DE UNA FILA ESPECIFICA

```
UPDATE ` Nombre Tabla `
SET ` Nombre Campo ` = 'Valor', ` Nombre Campo ` = ' Valor',` Nombre
Campo ` = ' Valor'
WHERE `id` = 'Numero Fila';
```

Nota: Siempre se incluye el campo id, a fin de identificar con un valor numérico una fila.

BORRAR UNA FILA

```
DELETE FROM ` Nombre Tabla `
WHERE `id`='Numero Fila';
```

SELECCIONAR DATOS DE UNA FILA

```
SELECT Nombre Campo, Nombre Campo
FROM Nombre Tabla where id = Numero Fila;
```

BUSCAR DATOS DENTRO DE UNA TABLA

SELECT Nombre Campo
FROM Nombre Tabla where Nombre Campo LIKE '%".Concepto de Búsqueda."%;

CONTAR REGISTROS TOTALES EN UN CAMPO

SELECT COUNT (Nombre Campo) FROM Nombre Tabla;

BACKUPS DE LA BASE DE DATOS:

Realiza siempre un respaldo de tu base de datos de manera periódica.
Da clic sobre el nombre de la base de datos en el menú del lado izquierdo.
Da clic en EXPORT.

Selecciona las tablas que desees respaldar.

Selecciona los botones de radio STRUCTURE y DATA

Selecciona los campos DROP TABLE Y ADD IF NOT EXITS

Selecciona ENCLOSE TABLE AND FIELNAMES WITH BACKQUOTES.

Selecciona los check boxes SAVE AS FILE y ZIPPED.

Al final solo haz clic en GO para exportar las tablas seleccionadas.

INTEGRANDO PHP Y MYSQL

En el capitulo anterior vimos cómo manejar una base de datos MySQL , como interactuar con ella usando las sentencias SQL, las cuales nos permitían seleccionar, actualizar, insertar y eliminar registros de nuestras tablas.
Ahora aprenderemos como usar estas sentencias pero usando PHP para por ejemplo hace un programa que me muestre mi lista de contactos en una pagina web.

CONECTAR A MYSQL DESDE PHP:

Para conectarnos a una base de datos desde php usaremos la funcion mysql_connect() ,veamos un ejemplo:

```
<?php
$Connec= mysql_connect("localhost", "root","root");
     echo “ya estas conectado”;
?>
```

Solo necesitamos especificar el Host o servidor, el usuario y si es que fuese necesario el password para poder validarnos en la base de datos. El resultado de la conexión es almacenado en la variable $Connec.

Ahora veamos como poder seleccionar una base de datos y enviar una consulta SQL para después mostrarla en php.

```
<?php
$Connec= mysql_connect("localhost", "root","root");
mysql_select_db("web", $Connec);
$result = mysql_query("SELECT * FROM agenda", $Connec);
echo "Nombre: ".mysql_result($result, 0, "nombre")."<br>";
echo "Dirección: ".mysql_result($result, 0, "direccion")."<br>";
echo "Teléfono :".mysql_result($result, 0, "telefono")."<br>";
echo "E-Mail :".mysql_result($result, 0, "email")."<br>";

?>
```

Con mysql_select_db() PHP le dice al servidor que en la conexión $Connecnos queremos conectar a la base de datos mydb. Podríamos establecer distintas conexiones a la BD en diferentes servidores, pero nos conformaremos con una.
La siguiente función mysql_query(), es la que hace el trabajo duro, usando el identificador de la conexión ($link), envía una instrucción SQL al servidor MySQL para que éste la procese. El resultado de ésta operación es almacenado en la variable $result.
Finalmente, mysql_result() es usado para mostrar los valores de los campos devueltos por la consulta ($result). En este ejemplo mostramos los valores del registro 0, que es el primer registro, y mostramos el valor de los campos especificados.

MOSTRAR TODOS LOS DATOS DE UNA CONSULTA SQL:

Ahora veremos como mostrar todos datos de una consulta.

```
<html>
<body>

<?php

$Connec= mysql_connect("localhost", "root","root");
mysql_select_db("web", $Connec);
$result = mysql_query("SELECT nombre, email FROM agenda", $Connec);

echo "<table border = '1'> \n";
echo "<tr><td>Nombre</td><td>E-Mail</td></tr> \n";

$total_rows = mysql_num_rows($result);

for($i=0;$i<=$total_rows;$i++)
{
        $nombre = mysql_result($result, $i, "nombre");
$email = mysql_result($result, $i, "email");

echo "<tr><td>$nombre</td><td>$email</td></tr> ";


}
echo "</table> ";
?>
 </body>
```



```
</html>
```

Como podemos observar hemos usado la funcion mysql_num_rows que devuelve el número de filas de un resultado asi en el ciclo FOR sabemos la cantidad de veces que se mostrara el registro. Nótese que reemplazamos el 0 de la funcion mysql_result por la variable $i para ir avanzando entre registros.

También se puede mostrar los resultados de una consulta usando otras funciones como veremos a continuación.

```
<html>
<body>

<?php
$Connec= mysql_connect("localhost", "root","root");
mysql_select_db("web", $Connec);

$result = mysql_query("SELECT nombre, email FROM agenda", $Connec);
echo "<table border = '1'> \n";
echo "<tr><td>Nombre</td><td>E-Mail</td></tr> \n";
while ($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.

```
<?php
      $Connec= mysql_connect("localhost", "root","root");
```

```
  mysql_select_db("web", $Connec);

$result = mysql_query("SELECT nombre, email FROM agenda", $Connec);
if ($row = mysql_fetch_array($result)){
   echo "<table border = '1'> \n";
   echo "<tr><td>Nombre</td><td>E-Mail</td></tr> \n";
   do {
      echo "<tr><td>".$row["nombre"]."</td><td>".$row["email"]."</td></tr>
\n";
   } while ($row = mysql_fetch_array($result));
   echo "</table> \n";
} else {
echo "¡ No se ha encontrado ningún registro !";
}
?>
```

Esencialmente, este script hace lo mismo que el anterior. Almacenamos en $row el registro actual con la función mysql_fetch_array() que hace exactamente lo mismo que mysql_fetch_row(), con la exepción que podemos referenciar a los campos por su nombre ($row["email"]), en vez de por un número.
Con la sentencia if/else, asignamos a $row el primer registro de la consulta, y en caso de no haber ninguno (else) mostramos un mensaje ("No se ha encontrado..."). Mientras que con la sentencia do/while, nos aseguramos que se nos muestren todos los registros devueltos por la consulta en caso de haber más de uno.
Hay que destacar la utilización del punto (.), como operador para concatenar cadenas.

AÑADIR REGISTROS A NUESTRA BASE DE DATOS:

Ahora veremos cómo podemos añadir nuevos registros a nuesta BD. La recogida de datos la vamos a hacer a traves de un interfaz de web. En primer lugar vamos a crear una página web con un simple formulario, con los campos que deseamos.
Formulario inicial añadir BD

```
<html>
<body>
<form method="post" action="insertar_reg.php">
Nombre   :<input type="Text" name="nombre"><br>
Dirección:<input type="Text" name="direccion"><br>
```

```
Teléfono :<input type="Text" name="telefono"><br>
E-mail   :<input type="Text" name="email"><br>
<input type="Submit" name="enviar" value="Aceptar información">
</form>
</body>
</html>
```

Hemos creado un formulario donde recoger los datos, y una vez introducidos ejecutamos un script llamado insertar_reg.php, pues veamos como es este script.

```
<html>
<body>
<?php
// process form
     $Connec= mysql_connect("localhost", "root", "root");
mysql_select_db("web", $Connec);
$sql = "INSERT INTO agenda (nombre, direccion, telefono, email) ".
  "VALUES ('$nombre', '$direccion', '$telefono', '$email')";
$result = mysql_query($sql,$Connec);
echo "¡Gracias! Hemos recibido sus datos.\n";
</body>
</html>
```

Como se puede ver, para introducir un nuevo registro, utilizamos la ya conocida función mysql_query(), la cuál también usamos para las consultas, y usaremos para las actualizaciones, es decir una señora función. Una cosa muy importante, para poder añadir o modificar registros debemos tener permiso para ello en el servidor MySQL, por eso en este caso me conecto como root, pero podría ser cualquier otro usuario.

MODIFICAR REGISTROS DE NUESTRA BASE DE DATOS:

Lo primero, es lo primero, para modificar hay que tener permiso para ello en el servidor de BD, el resto nos viene de corrido. Primero seleccionamos el registro que deseamos modificar, y luego, mandamos una consulta con las modificaciones, o ambas cosas a la vez. Suponemos que las modificaciones las recogemos de un formulario como el de la lección anterior.

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

```
<?php
if (isset($id)){
   // process form
   $Connec= mysql_connect("localhost", "root",”root”);
   mysql_select_db("web", $Connec);

   $sql = "SELECT * FROM agenda WHERE id = $id"
   $result = mysql_query($sql,$Connec);
   $sql = "UPDATE agenda SET nombre='$nombre',
    direccion='$direccion',".
      "telefono='$telefono', email='$email'";
   $result = mysql_query($sql);
}else{
   echo "Debe especificar un 'id'.\n";
}

</body>
</html>
```

También podría hacerlo de esta forma

```
<?php
if (isset($id)){
   // process form
        $Connec= mysql_connect("localhost", "root",”root”);
   mysql_select_db("web", $Connec);
   $sql = "UPDATE agenda SET nombre='$nombre', direccion='$direccion',".
      "telefono='$telefono', email='$email' WHERE id=$id";
   $result = mysql_query($sql,$Connec);
}else{
   echo "Debe especificar un 'id'.\n";
}
```


BORRAR REGISTROS DE NUESTRA BASE DE DATOS:

El proceso de borrar un registro es idéntico al de modificar, solo que en vez de utilizar UPDATE utilizamos DELETE en la sentencia SQL. Por tanto el script quedaría como sigue.

```
<html>
<body>

<?php
if (isset($id)){
   // process form
   $Connec= mysql_connect("localhost", "root",”root”);
   mysql_select_db("web", $Connec);
   $sql = "DELETE agenda WHERE id=$id")
   $result = mysql_query($sql,$Connec);
}else{
   echo "Debe especificar un 'id'.\n";
}

</body>
</html>
```


