Desarrollo Web con PHP y MySQL

Ing. Joel Gonzalez Estrada

php

MySQL

Índice

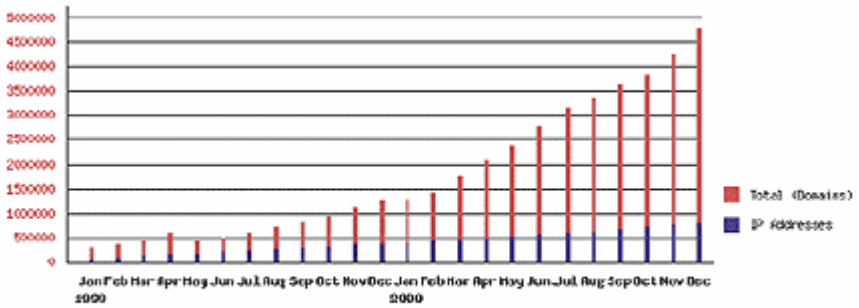
Conectándose y desconectándose al servidor MySQL
Creando y usando una base de datos
Visualización de las bases de datos existentes en el servidor MySQL
Selección de una base de datos
Creación de una base de datos
Creación de tablas
Ingreso de Datos a las tablas
Recuperación de la Información

PHP para bases de datos MySQL

Conectarse
Agregar registros
Modificar registros
Eliminar registros

CORTA HISTORIA DE PHP

PHP es un lenguaje creado por una gran comunidad de personas. El sistema fue desarrollado originalmente en el año 1994 por **Rasmus Lerdorf** como un CGI escrito en C que permitía la interpretación de un número limitado de comandos. El sistema fue denominado Personal Home Page Tools y adquirió relativo éxito gracias a que otras personas pidieron a Rasmus que les permitiese utilizar sus programas en sus propias páginas. Dada la aceptación del primer PHP y de manera adicional, su creador diseñó un sistema para procesar formularios al que le atribuyó el nombre de FI (Form Interpreter) y el conjunto de estas dos herramientas, sería la primera versión compacta del lenguaje: PHP/FI. La siguiente gran contribución al lenguaje se realizó a mediados del 97 cuando se volvió a programar el analizador sintáctico, se incluyeron nuevas funcionalidades como el soporte a nuevos protocolos de Internet y el soporte a la gran mayoría de las bases de datos comerciales. Todas estas mejoras sentaron las bases de PHP versión 3. Actualmente PHP se encuentra en su versión 4, que utiliza el motor Zend, desarrollado con mayor meditación para cubrir las necesidades actuales y solucionar algunos inconvenientes de la anterior versión. Algunas mejoras de esta nueva versión son su rapidez -gracias a que primero se compila y luego se ejecuta, mientras que antes se ejecutaba mientras se interpretaba elcódigo-, su mayor independencia del servidor web -creando versiones de PHP nativas para más plataformas- y un API más elaborado y con más funciones.



Gráfica del número de dominios y direcciones IP que utilizan PHP.
Estadística de Netcraft.

En el último año, el número de servidores que utilizan PHP se ha disparado, logrando situarse cerca de los 5 millones de sitios y 800.000 direcciones IP, lo que le ha convertido a PHP en una tecnología popular.

¿QUE ES PHP?

El lenguaje PHP es un lenguaje de programación de estilo clásico, es decir, es un lenguaje de programación con variables, sentencias condicionales, ciclos (bucles), funciones.... No es un lenguaje de marcado como podría ser HTML, XML o WML. Está más cercano a JavaScript o a C, para aquellos que conocen estos lenguajes.

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 pero igualmente podría ser una pagina WML.

Al ser PHP un lenguaje que se ejecuta en el servidor no es necesario que su navegador lo soporte, es independiente del browser, pero sin embargo para que las páginas PHP funcionen, el servidor donde están alojadas debe soportar PHP.

¿QUE NECESITA PARA QUE FUNCIONE PHP?

lo que necesita es lo siguiente:

- _ Versión compilada de PHP (http://www.php.net).
- _ Un servidor web (Apache, PWS, IIS, Etc.).
- _ Si desea manejar base de datos se recomienda Mysql Server (http://www.mysql.com).

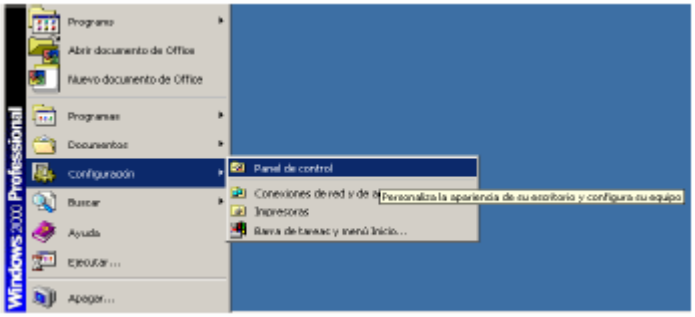
INICIAR CON PHP

- _ Instalar el servidor WEB.

Trabajaremos con sistemas operativos Microsoft Windows (2000 Professional y XP
Professional), por lo que vamos a instalar un IIS (versión 5 para w2k y 5.1 para wXP).

Para instalarlo tenemos que seguir los siguientes pasos:

- _ Ir al panel de control de Microsoft Windows©



- _ Seleccionar lo opción Agregar o quitar programas

- _ Seleccionar la opción de Agregar o quitar componentes y ahí elegir la opción Servicios de Internet Information Server (IIS)

- _ Tenemos que esperar a que se instale

- Después de estos pasos ya esta instalado nuestro servidor de web que necesitamos para ejecutar páginas de Internet.

La forma de saber si se ha instalado correctamente nuestro servidor es

tecleando en la barra de direcciones de nuestro navegador (Internet Explorer o Netscape Navigator) la siguiente dirección http://localhost y nos debe aparecer la siguiente página web:

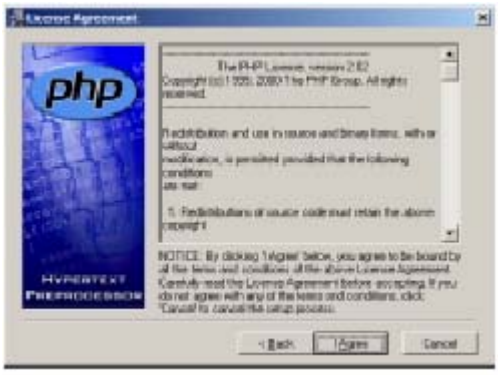
Instalar el modulo de PHP.

Los pasos para instalar el modulo de PHP son los siguientes:

- _ Al iniciar el programa de instalación veremos la siguiente pantalla.

- La siguiente pantalla es la de bienvenida, solo hay que presionar el botón de **Next >.**

- La pantalla que sigue es la licencia de PHP, si la aceptamos debemos de presionar **I Agree.**



Lo que sigue es elegir el tipo de instalación, cuenta con dos opciones, estándar y avanzado, para nuestras necesidades elegiremos la forma estándar, por lo tanto seleccionamos el botón de radio que dice **Standar**.

Ahora tendremos que elegir el directorio en el cual se instalará el PHP, el instalador trae por default el directorio C:\PHP, lo dejaremos así, por lo tanto presionamos **Next**.

Lo que nos pide la siguiente pantalla son los datos para el administrador del sistema por si falla algo en PHP, para este curso solo presionamos **Next**, pero

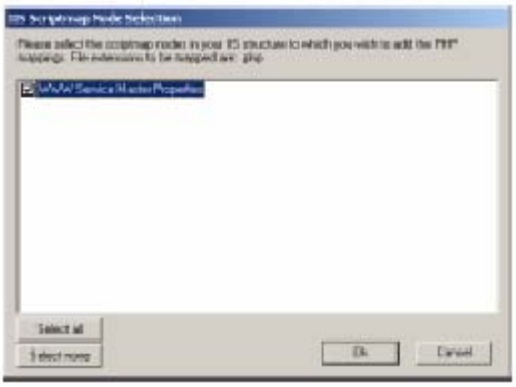
también puede introducir una dirección válida.

Ahora se nos pregunta el tipo de servidor que estamos utilizando, como ya había mencionado antes, asaremos IIS versión 5 y 5.1, por lo tanto seleccionamos el botón de radio que tiene la opción de **Microsoft IIS 4 or higher**.

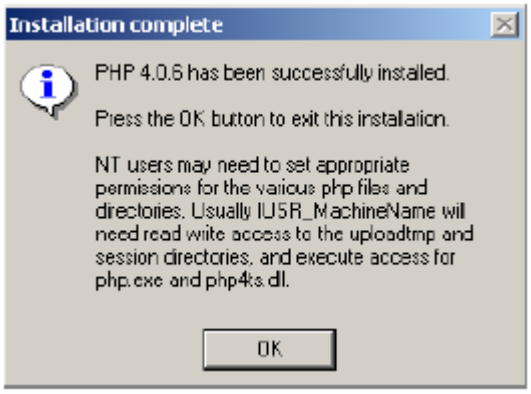
La siguiente pantalla nos avisa que ya esta listo para instalar PHP, así que solo presionamos **Next**.

Veremos como va el proceso de la instalación.

La siguiente pantalla lo que nos dice es que seleccionemos que parte del IIS queremos que ejecute PHP, como solo tenemos instalado el servicio de web, solo nos presenta esa opción, pero con eso es suficiente, así que seleccionamos el checkbox que dice **WWW Service Master Properties**.



Por ultimo nos dice que la instalación se ha completado y estamos listo para ejecutar PHP en nuestro servidor.



La forma de saber si se ha instalado correctamente el PHP en nuestro servidor haciendo un pequeño programita PHP, el mas básico que nos servirá de prueba para ver si se ha instalado correctamente, entonces abrimos un editor de texto cualquiera (con el cual haremos nuestros programas PHP, puede ser el NotePad), y escribimos las siguientes líneas:

```
<?
phpinfo();
?>
```

después de escribir estas líneas guardamos nuestro archivo en el directorio **c:\inetpub\wwroot\taller** el cual va a ser nuestro directorio de trabajo para este curso,
(*como información, el IIS utiliza como directorio raíz el directorio c:\inetpub\wwroot*), le
pondremos al archivo **prueba.php** ahora tecleamos en la barra de direcciones de nuestro

navegador (Internet Explorer o Netscape Navigator) la siguiente dirección http://localhost/taller/prueba.php y nos debe aparecer la siguiente página web:

Si no aparece esta página, entonces esta mal instalado el PHP y hay que volverlo a instalar, aunque hay algunas ocasiones que se necesita reiniciar el sistema para que funcione.

PROGRAMACION CON PHP

Ahora que ya tenemos instalado nuestro servidor y el modulo de PHP, podemos iniciar ha hacer programas PHP, pero ¿Cómo hacerlos?, un ejemplo nos aclarará las cosas:

```
<html>
<head>
<title>Ejemplo PHP</title>
</head>
<body>
<?php echo "Hola, este es un ejemplo con PHP"; ?>
</body>
</html>
```

Podemos ver que no es lo mismo que un script CGI escrito en otro lenguaje de programación como Perl o C -- En vez de escribir un programa con muchos comandos ara
crear una salida en HTML, escribimos el código HTML con cierto código PHP embebido (introducido) en el mismo, que producirá cierta salida (en nuestro ejemplo, producir un texto). El código PHP se incluye entre etiquetas especiales de comienzo y final que nos permitirán entrar y salir del modo PHP.

Pero cuales son estas etiquetas especiales de comienzo y final??, esto nos lleva a que hay 4 formas de salir de HTML y entrar en modo PHP, las cuales son las siguientes:

- <? echo (“Forma 1"); ?>
- <?php echo(“Forma 2"); ?>
- <script language="php"> echo (“Forma 3"); </script>
- <% echo("Etiquetas tipo ASP"); %>

SEPARACION DE INSTRUCCIONES

Las instrucciones se separan igual que en C o Pascal terminando cada sentencia con un punto y coma.
La etiqueta de cierre (?>) también implica el fin de la sentencia, así lo siguiente es equivalente:

```
<?php echo "Esto es una prueba"; ?>
<?php echo "Esto es una prueba" ?>
```

COMENTARIOS EN PHP

PHP soporta comentarios tipo 'C', 'C++' y Shell de Unix. Por ejemplo:

```
<?php
echo “Prueba"; // Comentario tipo C++ para una línea
```

```
?>

<?php
/*
Esto es un comentario multilínea
otra línea más de comentario
*/
echo "Esto es aún otra prueba"; ?>
<?php
echo "Prueba"; # Comentario tipo shell de Unix
?>
```

VARIABLES

Antes de ver como se utilizan las variables en PHP veremos una pequeña introducción a las variables, para los que no conozcan que es una variable.

Una variable consiste en un elemento al cual le damos un nombre y le atribuimos un determinado tipo de información. Las variables pueden ser consideradas como la base de la programación.

De este modo podríamos escribir en un lenguaje ficticio:
a="perro"
b="muerde"

La variable que nosotros llamamos "a" posee un elemento de información de tipo texto que es "perro". Asimismo, la variable "b" contiene el valor "muerde".

Podríamos definir una tercera variable que fuese la suma de estas dos:
c=a+b

Si introdujésemos una petición de impresión de esta variable en nuestro lenguaje
ficticio:

imprimir(c)
El resultado podría ser:

perro muerde
Podríamos de la misma forma trabajar con variables que contuviesen números y construir nuestro programa:

a=3
b=4
c=a+b

imprimir(c)

El resultado de nuestro programa sería:

7
En PHP las variables se representan como un signo de pesos seguido por el nombre de la variable. El nombre de la variable es sensible a minúsculas y mayúsculas.

```
$var = "Santana";
$Var = "Pedro";
echo "$var, $Var";
// produce la salida “Santana, Pedro"
```

VARIABLES VARIABLES

A veces es conveniente tener nombres de variables variables. Dicho de otro modo, son nombres de variables que se pueden establecer y usar dinámicamente. Una variable normal se establece con una sentencia como:

```
$a = "Hola";
```

Una variable variable toma el valor de una variable y lo trata como el nombre de una variable. En el ejemplo anterior, *Hola*, se puede usar como el nombre de una variable utilizando dos signos de peso. p.ej.
```
$$a = "mundo";
```

Probemos las siguientes sentencias:

```
echo "$a ${$a}";
echo "$a $Hola";
```

Los dos no regresarán “**Hola mundo**”

TIPOS DE DATOS

ENTEROS

Los enteros se puede especificar usando una de las siguientes sintaxis:

```
$a = 1234; # número decimal
$a = -123; # un número negativo
$a = 0123; # número octal (equivalente al 83 decimal)
$a = 0x12; # número hexadecimal (equivalente al 18 decimal)
```

NUMEROS ENTEROS FLOTANTES

Los números en punto flotante ("double") se pueden especificar utilizando

cualquiera de las siguientes sintaxis:
$a = 1.234;
$a = 1.2e3;

CADENAS

El único operador de cadenas que existen es el de concatenación, el **punto**. Pero no se preocupen, PHP dispone de toda una gama de funciones que nos permitirán trabajar cómodamente con las cadenas.

$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"

Si la cadena está encerrada entre dobles comillas ("), las variables que estén dentro de la cadena serán expandidas (sujetas a ciertas limitaciones de interpretación). Como en C y en Perl, el carácter de barra invertida ("\") se puede usar para especificar caracteres especiales:

CARACTERES PROTEGIDOS

\n Nueva línea
\r Retorno de carro
\t Tabulación horizontal
\\ Barra invertida
\$ Signo del dólar
\" Comillas dobles
\[0-7]{1,3} La secuencia de caracteres que coincida con la expresión regular es un carácter en notación octal
\x[0-9A-Fa La secuencia de caracteres que coincida con la expresión regular es f]{1,2} un carácter en notación hexadecimal

OPERADORES DE COMPARACION

$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

OPERADORES 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 $b Verdadero si sólo uno de los dos es verdadero
!$a Verdadero si $a es falso

OPERADORES DE ASIGNACIÓN

$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

Para demostrar el uso de algunos operadores mostraré algunos ejemplos en PHP:
El siguiente programa calcula el salario de un trabajador con un impuesto

```
<html>
<body>
<?php
$SalarioTrabajador = 3500;
$Impuesto = 20; // Porcentaje
$SueldoReal = $SalarioTrabajador - (($SalarioTrabajador /
100) * $Impuesto);
echo “Sueldo del trabajador sin impuesto:
$SalarioTrabajador<BR>”;
echo “Con el impuesto :$SueldoReal”;
?>
</body >
</html>
```

*Programa en PHP que calcula el área de un triangulo cuya formula es a=(b*h)/2*

```
<?php
$Base=15;
$Altura=12;
$Area = ($Base * $Altura)/2;
printf (“El area del triangulo es: $Area”);
?>
```

Programa que dados 2 números calcule la suma, resta, multiplicación, división, y modulo.

```
<?php
$Num1=8;
$Num2=5;
printf("La suma de $Num1 y $Num2 es: <b>%d</b><br>", $Num1 +
$Num2 );
printf("La resta de $Num1 y $Num2 es: <b>%d</b><br>", $Num1
- $Num2 );
printf("La multiplicación de $Num1 y $Num2 : <b>%d</b><br>",
$Num1 * $Num2 );
printf("La division de $Num1 y $Num2: <b>%0.2f</b><br>",
$Num1 / $Num2 );
printf("El modulo de $Num1 y $Num2 es <b>%0.1f</b><br>",
$Num1 % $Num2 );
?>
```

CONSTANTES

PHP define varias constantes y proporciona un mecanismo para definir más en tiempo de ejecución. Las constantes son como las variables, salvo por las dos circunstancias de que las constantes deben ser definidas usando la función **define()**, y que
no pueden ser redefinidas más tarde con otro valor.
Las constantes predefinidas (siempre disponibles) son:
__FILE__
El nombre del archivo de comandos que está siendo interpretado actualmente. Si se usa dentro de un archivo que ha sido incluido o requerido, entonces se da el nombre del archivo incluido, y no el nombre del archivo padre.
__LINE__
El número de línea dentro del archivo que está siendo interpretado en la actualidad. Si se usa dentro de un archivo incluido o requerido, entonces se da la posición dentro del archivo incluido.
PHP_VERSION
La cadena que representa la versión del analizador de PHP en uso en la actualidad.
PHP_OS
El nombre del sistema operativo en el cuál se ejecuta el analizador PHP.
TRUE
Valor verdadero.
FALSE
Valor falso.

E_ERROR
Denota un error distinto de un error de interpretación del cual no es posible recuperarse.
E_WARNING
Denota una condición donde PHP reconoce que hay algo erróneo, pero continuará de todas formas; pueden ser capturados por el propio archivo de comandos.
E_PARSE
El interprete encontró sintaxis inválida en el archivo de comandos. La recuperación no es posible.
E_NOTICE
Ocurrió algo que pudo ser o no un error. La ejecución continúa. Los ejemplos incluyen usar una cadena sin comillas como un índice "hash", o acceder a una variable que no ha sido inicializada.
Las constantes E_* se usan típicamente con la función **error_reporting()** para
configurar el nivel de informes de error.
Se pueden definir constantes adicionales usando la función **define()**.
Nótese que son constantes, con una constante sólo se pueden representar datos escalares válidos.
Veremos un ejemplo del uso de estas constantes:

```
<?php
function report_error($archivo, $linea, $mensaje) {
echo "Un error ocurrió en $archivo en la línea $linea:
$mensaje.";
}
report_error(__FILE__,__LINE__, "Algo esta mal!");
?>
```

Ahora veremos como definir nuestras propias constantes:

```
<?php
define("CONSTANTE", "Hola mundo.");
echo CONSTANTE; // muestra "Hola mundo."
?>
```

SENTENCIAS DE CONTROL

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 que

la condición que evaluemos ha de estar encerrada entre paréntesis (esto es aplicable a todas
la sentencias de control).

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

Ahora realizaremos un ejemplo con mas con IF el cual consistirá en un pequeño juego de adivinanzas el cual necesitará de dos archivos *adivina.htm* y *adivina.php*, en este ejemplo se utilizará un formulario, lo cual aun no hemos visto, se usa solo para muestra, por el momento no es muy importante conocer esa teoría sino un poco mas adelante.

adivina.htm

```
<HTML>
<BODY>
<FORM METHOD=GET ACTION="adivina.php">
En que numero del 1 al 10 estoy pensando?
<INPUT NAME="adivina" TYPE="Text">
<BR>
<BR>
<INPUT TYPE=SUBMIT>
</FORM>
</BODY>
</HTML>
```

adivina.php

```
<HTML>
<HEAD></HEAD>
<BODY>
<?php
srand((double)microtime()*1000000);
$Numero = rand(1,10);
if ($adivina > $Numero) {
echo "Fue muy grande"; echo "<BR>Yo pensé el número
$Numero. Lo siento no ";
}
if ($adivina < $Numero) {
echo "Fue muy pequeño"; echo "<BR>Yo pensé el número
```

```
$Numero. Lo siento no ";
}
?>

GANASTE
</BODY>
</HTML>
```

if...elseif...else

La sentencia IF...ELSEIF...ELSE permite ejecutar 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";
}
?>
```

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 ciclo 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 ciclo for no es estrictamente necesario, cualquier ciclo for puede ser sustituido
fácilmente por otro while. Sin embargo, el ciclo 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 ciclo 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
}
}
?>
```

A continuación muestro las 4 formas en que se puede usar el ciclo for.

```
/* ejemplo 1 */
for ($i = 1; $i <= 10; $i++) {
print $i;
}
/* ejemplo 2 */
for ($i = 1;;$i++) {
if ($i > 10) {
break;
}
}
print $i;
}
/* ejemplo 3 */
$i = 1;
for (;;) {
```

```
if ($i > 10) {
break;
}
print $i;
$i++;
}
/* ejemplo 4 */
for ($i = 1; $i <= 10; print $i, $i++) ;
```

Hay que realizar los siguientes programas con ciclos.

· Imprima los números del 1 al 100

```
for ($i = 1; $i <= 100; print $i."<br>", $i++) ;
```

· Imprima los números pares del 1 al 100

```
for ($i = 2; $i <= 100; print $i."<br>", $i=$i+2) ;
```

· Un programa que le des un número y obtenga su tabla de multiplicar (*tablas.php*)

tablas.php

```
<?php
$numero=4;
for ($i=1; $i<=10; $i++)
{
echo $i." x ".$numero." = ".$i*$numero."<br>";
}
?>
```

VECTORES (TABLAS)

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[] = "México";
$ciudad[] = "Roma";
$ciudad[] = "Sevilla";
$ciudad[] = "Londres";
print ("yo vivo en " . $ciudad[1] . "<BR>\n");
?>
```

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

```
<?php
$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");
}
?>
```

Sino se especifica, el primer índice es el **cero**, pero podemos utilizar el operador **=>**
para especificar el índice 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,

```
$visitas = array("codigo">$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 **siguiente**
prev - mueve el puntero al elemento **anterior**
count - devuelve el número de elementos de una tabla.

Veamos un ejemplo de las funciones anteriores:

```
<?php
$semana = array("lunes", "martes", "miércoles", "jueves",
"viernes", "sábado", "domingo");
echo count($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 enredoso, 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()**.

Tablas multidimensionales

Las tablas multidimensionales son simplemente tablas en las cuales cada elemento
es a su vez otra tabla.

```
<?php
$calendario[] = array (1, "enero", 31);
$calendario[] = array (2, "febrero", 28);
$calendario[] = array (3, "marzo", 31);
$calendario[] = array (4, "abril", 30);
$calendario[] = array (5, "mayo", 31);
while (list($clave, $valor ) = each($calendario)){
{
$cadena = $valor[1];
$cadena .= " es el mes número " . $valor[0];
$cadena .= "y tiene " . $varlor[2] . " días<BR>";
echo $cadena;
}
?>
```

La función **list()** es más bien un operador de asignación, lo que hace es asignar valores a unas lista de variables. En este caso los valores son extraídos de una tabla por la
función **each()**.

FORMULARIOS

Los Formularios no forman parte de PHP, sino del lenguaje estándar de Internet, HTML, pero como éstos van a aparecer muchas veces durante el curso, vamos a dedicar esta algunas líneas a ellos.

Todo formulario comienza con la etiqueta **<FORM ACTION="lo_que_sea.php" METHOD="post/get">**. Con **ACTION** indicamos el script que va procesar la información que recogemos en el formulario, mientras que **METHOD** nos indica si el usuario del formulario va ha enviar datos (**post**) o recogerlos (**get**). La etiqueta **</FORM>**
indica el final del formulario.

A partir de la etiqueta **<FORM>** vienen los campos de entrada de datos que pueden
ser:

Botón de comando:

```
<input type="submit" value="enviar" name="enviar">
```

Cuadro de texto:

```
<input type="text" name="nombre" size="20" value="jose">
```

Veamos un ejemplo con PHP:

Las siguientes dos paginas tienen el objetivo de preguntar cual es tu equipo de fútbol
favorito y desplegar en otra pagina el seleccionado (*equipo.htm* y *equipo.php*).

equipo.htm

```
<html>
<title>Equipo Favorito</title>
<body>
<form method=post ACTION="equipo.php">
Cual es tu equipo de fútbol favorito ?
<input name="equipo" type "TEXT">
<br>
<br>
<input type=submit>
</form>
</body>
<html>

<html>
<body>
Tu equipo favorito es:
<?php Echo "<h1><B>$equipo</B></h1>"; ?>
</body>
</html>
```

Hay que poner especial atención en el parámetro name de un elemento del formulario ya que es el mismo nombre con el que se le referenciará en php, como pudimos ver en el ejemplo anterior el elemento <input name="***equipo***" type "TEXT"> lo manejamos en php como ***$equipo***, así es con todos los elementos de formularios.

Cuadro de texto con barras de desplazamiento:

```
<textarea rows="5" name="descripcion" cols="20">Es de color
rojo</textarea>
```

Ahora veamos un ejemplo con PHP:
Programa PHP que pide WebSites favoritos y los muestra como salida (*sites.htm* y *sites.php*).

sites.htm

```
<html>
<title>Web Sites Favoritos</title>
<body>
<form method=POST ACTION="sites.php">
Menciónname algunos de tus WebSites Favoritos:
<br>
<textarea name="websites" cols="50" rows="5">
```

```
http://
http://
http://
http://
</textarea>
<br>
<br>
<input type=submit>
</form>
</body>
<html>
```

sites.php

```
<html>
<body>
Tus webs favoritos son:<br>


<?php Echo “<h3><B>$websites</B></h3>”; ?>
</body>
</html>
```

Casilla de verificación o checkbox:

```
<input type="checkbox" name="cambiar" value="ON">
```

Ahora veamos un ejemplo con PHP:
Programa que pregunta lo que haces al levantarte y lo despliega como salida (*checkboxes.htm* y *checkboxes.php*).

checkboxes.htm

```
<HTML>
<HEAD></HEAD>
<BODY>
<FORM METHOD=POST ACTION=”checkboxes.php”>
Qué haces en cuanto te levantas?<br><br>
Lavarme la cara<INPUT NAME=”sel1” TYPE=”Checkbox”
VALUE=”Lavarse la Cara”><BR>
Asearse la boca<INPUT NAME=”sel2” TYPE=”Checkbox”
VALUE=”Asearse los dientes”><BR>
Desayunar<INPUT NAME=”sel3” TYPE=”Checkbox” VALUE=”Desayunar”
><BR>
<BR>
<INPUT TYPE=SUBMIT>
</FORM>
</BODY>
</HTML>
```

checkboxes.php

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

```
<?php
if (isset($sel1))
Echo “$sel1 <br>”;
if (isset($sel2))
Echo “$sel2 <br>”;
if (isset($sel3))
Echo “$sel3 <br>”;
?>
</body>
</html>
```

Botón de radio u opción:

```
<input type="radio" value="azul" checked name="color">
```

Ahora veamos un ejemplo con PHP:
Programa que nos presenta una suma, nosotros tenemos que elegir la respuesta entre
tres opciones posibles y la salida nos muestra la opción que elegimos (*radio.htm* y
radio.php).

radio.htm

```
<HTML>
<BODY>
<FORM METHOD=GET ACTION=”radio.php”>
Cuantos son 2 + 2?
<BR>
<BR>
<INPUT NAME=”Resp” TYPE=”Radio” VALUE=”44”>44
<BR>
<INPUT NAME=”Resp” TYPE=”Radio” VALUE=”22”>22
<BR>
<INPUT NAME=”Resp” TYPE=”Radio” VALUE=”4”>4
<BR>
<BR>
<INPUT TYPE=SUBMIT>
</FORM>
</BODY>
```

radio.php

```
<HTML>
<BODY>
<?php Echo “seleccionaste $Resp”;?>
</BODY>
</HTML>
```

Menú desplegable:

```
<select size="1” class="codigo"><select size="1" name="dia">
<option selected value="lunes">lunes</option>
<option>martes</option>
```

```
<option value="miercoles">miercoles</option>
</select>
```

Ahora veamos un ejemplo en PHP:

Programa que pregunta opciones para armar una computadora y despliega las opciones elegidas (*lista.htm* y *lista.php*).

lista.htm

```
<HTML>
<HEAD></HEAD>
<BODY>
<FORM METHOD=GET ACTION="lista.php">
Elije la computadora a comprar
<BR>
<BR>
<SELECT NAME="compu">
<OPTION>Pentium</OPTION>
<OPTION>Celeron</OPTION>
<OPTION>K6</OPTION>
<OPTION>MAC</OPTION>
</SELECT>
<BR>
<BR>
Selecciona los dispositivos de la computadora?
<BR>
<BR>
<SELECT NAME="dispo[]" MULTIPLE>
<OPTION>KIT MULTIMEDIA</OPTION>
<OPTION>QUEMADORA</OPTION>
<OPTION>WEB CAM</OPTION>
<OPTION>MICROFONO</OPTION>
</SELECT>
<BR>
<BR>
<INPUT TYPE=SUBMIT>
</FORM>
</BODY>
</HTML>
```

lista.php

```
<HTML>
<BODY>
<?php
Echo "Seleccionaste una computadora: <B>$compu</B>
con:<br>";
Echo "$dispo[0]<br>";
Echo "$dispo[1]<br>";
Echo "$dispo[2]<br>";
Echo "$dispo[3]<br>";
?>
```

```
</BODY>
</HTML>
```

Campo oculto:

```
<input type="hidden" name="edad" value="55">
```

Este último tipo de campo resulta especialmente útil cuando queremos pasar datos
ocultos en un formulario.
Ahora pasemos a ver ejemplos que necesitan mas código PHP, como son ciclos y
arrays, implementándolos en conjunto con los formularios.
Para el uso del FOR un programita para créditos bancarios (*banco.htm* y *banco.php*).
banco.htm

```
<HTML>
<HEAD></HEAD>
<BODY>
<B>Crédito bancario</B>
<FORM METHOD=POST ACTION="banco.php">
<BR>
Cual de estos paquetes te interesa tomar?<BR><BR>
<INPUT NAME="valor" TYPE="Radio" VALUE=1000>Nuestro paquete
de $1,000 Con el 5.0% interes
<BR>
<INPUT NAME="valor" TYPE="Radio" VALUE=5000>Nuestro paquete
de $5,000 Con el 6.5% interes
<BR>
<INPUT NAME="valor" TYPE="Radio" VALUE=10000>Nuestro paquete
de $10,000 Con el 8.0% interes
<BR>
<BR>
Cuanto es lo que deseas pagar al mes ?
<INPUT NAME=pagomes TYPE=Text SIZE=5>
<BR>
<BR>
<INPUT TYPE=SUBMIT VALUE="Pulse aquí para calcular">
</FORM>
</BODY>
</HTML>
```

banco.php

```
<HTML>
<HEAD></HEAD>
<BODY>
<?php
$Duracion=0;
switch ($valor) {
case 1000:
```

```
$Interes = 5;
break;

case 5000:
$Interes = 6.5;
break;
case 10000:
$Interes = 8;
break;
default:
echo "No seleccionaste ningun paquete favor de
presionar el boton back y seleccionar alguno";
exit;
}
while ($valor > 0)
{
$Duracion = $Duracion + 1;
$Mensualmente = $pagomes - ($valor * $Interes/100);
if ($Mensualmente<=0)
{
echo "Tu necesitas hacer pagos mas grandes!";
exit;
}
$valor = $valor - $Mensualmente;
}
echo "La duracion es de: $Duracion meses con un
porcentaje de intereses del $Interes.";
?>
</BODY>
</HTML>
```

El siguiente programa demuestra como se pueden trabajar los array como elementos
hash *(estados.php* y *capital.php)*.

estados.php

```
<html>
<head>
<title>Estados de México</title>
</head>
<body bgcolor="#FFFFFF">
De que estado te gustaría conocer su capital?
<?
$Estados=array(1=>"Colima","Jalisco","Sinaloa");
echo "<form method=post action='capital.php'>";
echo "<select name='estado'>";
for ($counter=1; $counter<4; $counter++)
echo "<option value=$counter>$Estados[$counter]</option>";
```

```
echo "</select><br><br>";
echo "<input type=submit>";
echo "</form>";
?></body>
</html>
```

capital.php

```
<html>
<head>
<title>Capitales</title>
</head>
<body bgcolor="#FFFFFF">
<?php
$Capital=array(1=>"Colima","Guadalajara","Culiacan");
for ($counter=0;$counter<4;$counter++)
{
if ($counter==$estado)
{
echo "la capital del estado elegido es
&Capital[$counter]";
}
}
?>
</body>
</html>
```

Como podemos observar los dos archivos tienen extensión .php, esto se debe a que es necesario ejecutar código php en los dos, para poder formar dinámicamente las dos páginas.
Para tener más material con formularios realizaremos un programa PHP que contenga varios elementos de formulario juntos:
Desarrollar un programa en PHP que pida el Nombre (textbox), el apellido (textbox), la edad (textbox), domicilio (text area), seleccione el rango de sueldo que le gustaría ganar (listbox) y que seleccione como considera así mismo su desempeño laboral (radio button). El programa no aceptará curriculums que elijan un sueldo muy bajo, ni un sueldo extremo, ni tampoco si se considera pésimo (*minicu.htm y minicu.php)*.

minicu.htm

```
<HTML><HEAD></HEAD><BODY><B>Minicurriculum </B>
<FORM METHOD=POST ACTION="minicu.php">
Nombres:
<INPUT NAME="nombres" TYPE="Text">
Apellidos:
<INPUT NAME="apellidos" TYPE="Text">
Edad:
<INPUT NAME="edad" TYPE="Text"SIZE="3">
```

```
<BR>
<BR>
Domicilio:
<TEXTAREA NAME="Domicilio" ROWS=4 COLS=40>
</TEXTAREA>
<BR>
<BR>
Que salario deseas ganar?

<SELECT NAME="Salario">
<OPTION VALUE=0> Menos de $1000</OPTION>
<OPTION VALUE=1000>Entre $1,000 y $3,000</OPTION>
<OPTION VALUE=3000>Entre $3,000 y $5,000</OPTION>
<OPTION VALUE=5000>mas de $5,000</OPTION>
</SELECT>
<BR>
<BR>
Como consideras tu desempeño?<BR><BR>
<INPUT NAME="Desempe" TYPE="Radio" VALUE=0>Pesimo
<BR>
<INPUT NAME="Desempe" TYPE="Radio" VALUE=5>Regular
<BR>
<INPUT NAME="Desempe" TYPE="Radio" VALUE=10>Excelente
<BR>
<BR>
<INPUT TYPE=SUBMIT VALUE="Presione cuando este listo(a)">
<INPUT TYPE=RESET VALUE="Borra todo">
</FORM>
</BODY>
</HTML>
```

minicu.php

```
<HTML>
<BODY>
Sr(a) <?php Echo "$nombres $apellidos ";?>
en base a su edad (<?php Echo "$edad";?>) y sus aspiraciones <br>
económicas y su auto estimación hemos determinado que:<br>
<?php
echo "Salario deseado: $Salario <br>" ;
echo "Su desempeño elegido: $Desempe <br>";
if ((($Salario==0) OR ($Salario ==5000)) OR ($Desempe ==0))
{
Echo "Usted no cumple los requisitos para ser parte de
nuestra empresa";
}
else
{
Echo "Usted cumple satisfactoriamente nuestras
```

```
aspiraciones";
}
?>
</BODY>
</HTML>
```

BASES DE DATOS

Hasta ahora hemos visto la funciones básicas de PHP, lo que viene a continuación ya es sobre la forma en como podemos acceder a bases de datos, veremos como acceder a ellas en dos sistemas de bases de datos diferentes, los cuales son MySQL Server y Microsoft Access.
El primer sistema que veremos es MySQL Server, pero antes de ver código PHP, aprenderemos un poco de este sistema, para los ejemplos en PHP ya va a ser necesario que
tengamos una base de datos creada por lo tanto pondremos atención en las formas en que
estas se crean.

QUE ES MYSQL

Administrador de Base de Datos

Una base de datos es un conjunto de datos estructurados. Esto podría ser cualquier cosa, desde una simple lista de compras hasta una galería de pinturas o la gran cantidad de información que se maneja en una corporación. Para agregar, acceder y procesar los datos almacenados en una base de datos computacional, se necesita un sistema administrador de base de datos tal como MySQL. Además los computadores son muy buenos manejando grandes cantidades de datos, el administrador de base de datos juega un rol central en la computación, ya sea como utilidad autónoma o parte de otra aplicación.

Sistema administrador de base de datos relacionales

Una base de datos relacional almacena datos en tablas separadas, más bien colocando todos los datos en un gran almacén. Esto agrega velocidad y flexibilidad. Las tablas son enlazadas por relaciones definidas haciendo posible combinar datos desde varias tablas solicitadas. El SQL forma parte de MySQL, conocido como Lenguaje de Consultas Estructurado, es el lenguaje estandarizado más común usado para acceder base de datos.

Software de Fuente Abierta

Fuente abierta significa que es posible que pueda ser usado y modificado por cualquiera. Alguien puede bajar MySQL desde Internet y usar éste sin tener que pagar nada. Alguien puede estudiar el código fuente y cambiarlo de acuerdo a sus necesidades. MySQL usa el GPL de la GNU, para definir lo que se puede hacer con el software en diferentes situaciones.

Uso de MySQL

MySQL es muy rápido, seguro y fácil de usar. MySQL también ha desarrollado un conjunto de características muy prácticas, en estrecha cooperación con otros usuarios. MySQL fue desarrollado para manejar grandes bases de datos mucho más rápido que las soluciones existentes y ha sido usado exitosamente en ambientes de producción con altas

demandas, por varios años. Aunque está bajo un desarrollo constante, MySQL siempre ofrece conjunto de funciones muy poderoso y eficiente. La conectividad, velocidad y seguridad hace de MySQL una suite poderosa para acceder a bases de datos en Internet.

CARACTERISTICAS TECNICAS DE MYSQL

Características Técnicas de MySQL

MySQL es un sistema Cliente/Servidor que consta de un servidor SQL multi-hilo
que soporta diferentes backends, variados programas cliente y de librerías, administrador de herramientas y un programa de interface.
MySQL contribución para muchos de los software disponibles
Es mucho más probable que se encuentre que diversas aplicaciones ya soportan
MySQL. Los valores centrales de MySQL son :

- La mejor y más usada base de datos en el mundo.
- Disponible y Accesible para todos
- Fácil de usar
- Se está perfeccionando continuamente mientras permanece rápida y segura.
- Divertida para usar y perfeccionar.
- Libre de molestias.

CARACTERISTICAS PRINCIPALES DE MYSQL

A continuación se describen algunas de las características más importantes de MySQL:

- Escrito en C y C++, testado con GCC 2.7.2.1. Usa GNU autoconf para portabilidad.
- Clientes C, C++, Eiffel,PHP,Python,JAVA, Perl, TCL.
- Multiproceso, es decir puede usar varias CPU si éstas están disponibles.
- Puede trabajar en distintas plataformas y S.O. distintos.
- Sistema de contraseñas y privilegios muy flexible y segura.
- Todas la palabras de paso viajan encriptadas en la red.
- Registros de longitud fija y variable.
- 16 índices por tabla, cada índice puede estar compuesto de 1 a 15 columnas o partes de ellas con una longitud máxima de 127 bytes.

- Todas las columnas pueden tener valores por defecto.
- Utilidad *Isamchk* para chequear, optimizar y reparar tablas.
- Todos los datos están grabados en formato ISO8859_1.
- Los clientes usan TCP o UNIX Socket para conectarse al servidor.
- Todos los comandos tienen -help o -? Para las ayudas.
- Soporta diversos tipos de columnas como enteros de 1, 2, 3, 4, y 8 bytes, coma
- flotante, doble precisión, carácter, fechas, enumerados, etc.
- ODBC para Windows 95 (con fuentes), se puede utilizar ACCESS para conectar
- con el servidor.
- Muy rápida usando joins, optimizada para un barrido multi-joins.

Todas las funciones y operadores soportan en el **SELECT** y **WHERE** como partes
de consultas. Ejemplo:

- mysql> SELECT CONCAT(nombre," ",apellido) FROM nombre_tabla WHERE ingreso >10000 AND edad >30

Todas las clausulas SQL soportan **GROUP BY** y **ORDER BY**.

INSTALANDO MYSQL

Bien, ya conocemos que es MySQL, ahora procedamos a instalarlo en nuestro servidor para poder realizar los programas PHP con bases de datos MySQL. Al iniciar el instalador veremos la pantalla de “preparándose para instalar”, en esta solo hay que esperar que termine el 100%.

Después tenemos la pantalla de bienvenida, solo hay que dar clic en **Next**.

La siguiente interfaz nos muestra la información de la versión de MySQL que estamos instalando.

Ahora tendremos la ventana de donde queremos instalarlo, para evitar complicaciones con archivos de configuración, presionemos **Next** y se instalará en C:\mysql.

Hay que elegir el tipo de instalación, elijamos ***Typical***.

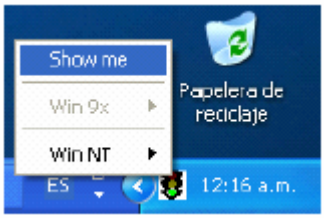
Esperamos un poco a que se instale.

Cuando termine hay que dar clic en **Finish**.

Después de estos sencillos pasos ya tenemos instalado MySQL Server, ahora lo que
sigue es iniciarlo, debemos ir a la siguiente ruta en nuestro disco duro ***C:\mysql\bin*** y
ejecutar el programa ***winmysqladmin.exe.***

Cuando este en ejecución WinMySQLadmin en su primera ocasión pedirá un nombre de usuario y contraseña para el administrador del sistema de base de datos.

Ahora si ya lo tendremos ejecutándose siempre que el sistema inicie, y estará en la barra de inicio del sistema con un icono de un semáforo, hay que darle clic en el icono para que nos muestre sus opciones.



Al darle clic en Show Me aparecerá el administrador de MySQL, desde ahí podemos ver todas los componentes del sistema de bases de datos.

Con esto es suficiente sobre la instalación de MySQL Server, ahora veamos como crear bases de datos en el.

CONECTANDOSE Y DESCONECTANDOSE

Para conectarse al servidor, generalmente se necesita proveer a **MySQL** un nombre de usuario, y un password. Si el servidor corre en una máquina distinta a la que se está utilizando se debe especificar el nombre del host (máquina). Cuando se conocen todos estos parámetros la conexión se realiza de la siguiente manera:

shell> mysql -h host -u user -p
Enter password:*********
Si la conexión se realizó con éxito, *mysql* despliega el siguiente mensaje:

Welcome to the MySQL monitor. Commands end with; or\g.
Your MySQL connection id is 459 to server version: 3.22.20a-log
Type "help" for help.
mysql>

El prompt indica que *mysql* está listo para recibir los comandos que ingrese el usuario. Algunas instalaciones de **MySQL** admiten usuarios *anonymous* (sin nombre) cuando el servidor corre en el host local. En este caso, se debe invocar a el servidor *mysql* sin ninguna opción:
shell>mysql
Una vez que se ha realizado la conexión con éxito, para desconectarse al servidor en cualquiera de los dos casos anteriores se debe escribir **QUIT** o **control-D**.

CREANDO Y USANDO UNA BASE DE DATOS

Visualización de las bases de datos existentes en el servidor MySQL
Antes de crear una base de datos, se debe conocer que base de datos existen actualmente en el servidor, para ello se utiliza el comando SHOW, de la siguiente manera:
mysql> SHOW DATABASES;

```
+----------+
| Database |
+----------+
| mysql    |
| test     |
| tmp      |
+----------+
```

Esta lista probablemente no es igual en todas las máquinas, pero las bases de datos **mysql** y **test** están siempre entre ellas. La base de datos **mysql** se requiere porque en ella se describe la información de los privilegios de acceso a los usuarios. La base de datos **test** proporciona el espacio de trabajo para los usuarios.

Selección de una base de datos
Para seleccionar o acceder a una base de datos determinada se utiliza el comando
USE:
mysql> USE test
Database changed
Una vez, que se ha realizado la conexión con éxito se puede comenzar a trabajar con
la base de datos, pero siempre y cuando se tengan los permisos adecuados. Si no se tienen
los permisos el administrador debe darle los permisos al usuario para poder trabajar, esto se realiza con la ejecución del siguiente comando:

mysql> GRANT ALL ON nombre_database.* TO nombre_usuario;

Creación de una base de datos

Para crear una base de datos se debe tener permiso para poder crear base de datos en el servidor MySQL , si se tiene el permiso entonces la sentencia a seguir es:
mysql> CREATE DATABASE nombre_database;
Bajo Unix, los nombres de las bases de datos y de las tablas son sensibles, esto quiere decir que se hace diferencia entre minúsculas y mayúsculas, así es que para referirse a una base de datos determinada hay que llamarla tal como se le nombro cuando fue creada.

Creación de tablas

Para crear las tablas que va a contener la base de datos, se realiza de la siguiente forma:
mysql> CREATE TABLE nombre_tabla(campo_1 tipo(tamaño), campo_2 tipo(tamaño),...,campo_n tipo(tamaño));
El *campo* indica el nombre de la columna y *tipo(tamaño)* específica el tipo de dato y el espacio que se va a conservar para cada dato almacenado en esa columna.
Ejemplo:
codigo int(5), nombre char(25), fecha date, etc.. Cuando se trata de fechas no se específica el tamaño, puesto que ya está determinado. Para visualizar las tablas que tiene una base de datos se usa el mismo comando utilizado para ver las bases de datos, pero con la diferencia de que en vez de **database** se coloca **tables**, es decir:
mysql> SHOW TABLES;
Para verificar que la tabla ha sido creada de la forma indicada, se usa el comando DESCRIBE. Ejemplo: Se va a crear una tabla llamada **clientes**, de la siguiente forma:
mysql> CREATE TABLE clientes(rut char(8),nombre char(25),
direccion char(50), telefono int(10));
mysql> DESCRIBE clientes;

```
+----------+-----------+------+-----+---------+-------+
| Field    | Type      | Null | Key | Default | Extra |
+----------+-----------+------+-----+---------+-------+
| rut      | char(12)  | YES  |     | NULL    |       |
| nombre   | char(25)  | YES  |     | NULL    |       |
| direccion| char(50)  | YES  |     | NULL    |       |
| telefono | int(10)   | YES  |     | NULL    |       |
+----------+-----------+------+-----+---------+-------+
```

Esto es muy útil cuando se olvida el nombre o tipo de una columna. El *Field* indica el nombre de la columna, *Type* es el tipo de dato que acepta esa columna, *Null* indica si la columna puede contener valores NULL, *Key* indica la clave por la cual la columna se va a indexar y *Default* específica el valor por defecto que tiene la columna.

Ingreso de Datos a las tablas

Para ingresar información a una tabla se puede hacer básicamente de dos maneras.
La primera se utiliza cuando se tiene mucha información a ingresar de una sola vez, entonces es conveniente almacenar esta información en un archivo de texto, es decir, *.txt*.
Una vez que se tiene este archivo, se procede de la siguiente forma:
mysql> LOAD DATA LOCAL INFILE "nombre_archivo.txt" INTO TABLE nombre_tabla; Para el caso que se desee ingresar un solo registro, entonces la sentencia a seguir es:
mysql> INSERT INTO nombre_tabla VALUES
(``valor_1'',"valor_2",...., Los datos a ingresar se separan por comas y van entre comillas. Estos datos indican los valores que va a tomar cada una de las columnas, según el orden en que fueron creadas.
En el caso que se quiera ingresar un valor NULL no es necesario las comillas, sólo se coloca NULL.

Recuperación de la Información

Para recuperar la información que está contenida en una tabla, la sentencia general a seguir es:
mysql> SELECT qué_es_lo_que_se_desea_ver FROM nombre_tabla WHERE condiciones_a_satisfacer;
Para los casos en que, se requiera:
o Ver o seleccionar toda la información de una tabla:
o mysql> SELECT * FROM nombre_tabla;
o Seleccionar filas en particular:
o mysql> SELECT * FROM nombre_tabla WHERE nombre_columna="lo que se desee buscar"
o Seleccionar columnas en particular:
o mysql> SELECT nombre_columna_1, nombre_columna_n FROM nombre_tabla;
Esto es conveniente cuando no se desea ver toda la fila o registro, entonces solo se seleccionan aquellas columnas en las que se esté interesado.

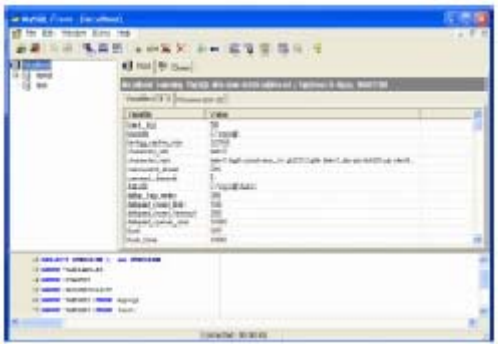
MYSQL FRONT

Ya vimos como crear bases de datos y tablas de la manera tradicional de MySQL, pero como podemos ver es algo complejo, y como ya estamos acostumbrados a interfaces gráficas (GUI por sus siglas en ingles), les mostraré como crear bases de datos de una manera completamente visual para no tener que tocar línea de comandos de MySQL, para esto utilizaremos el software **MySQL Front** desarrollado por Ansgar Becker con correo electrónico chef@anse.de y dirección de su página http://my.anse.de/forum.php en Alemania, aunque yo personalmente he intentado entrar a esa dirección y no he podido, pero son los datos que trae el programa.
Veamos como usarlo:
La primera vez que lo corremos no presentara un formulario en blanco y sin

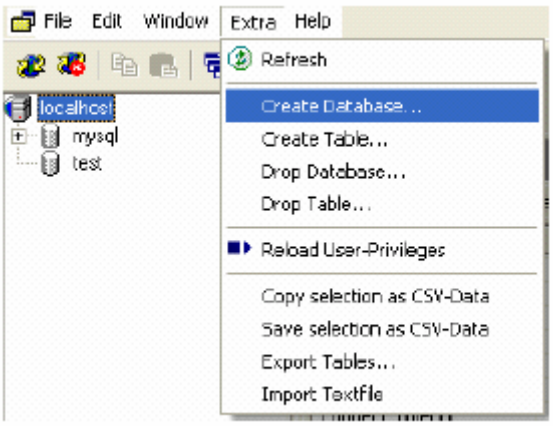
forma de poder ingresar datos, es necesario presionar el botón **New**, para habilitarlo.

Una vez presionado New aparecen unos datos ya predefinidos, lo único que tenemos que cambiar es nuestro usuario y contraseña.

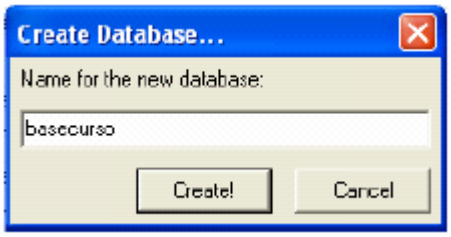
Al momento de entrar nos mostrará esta interfaz.



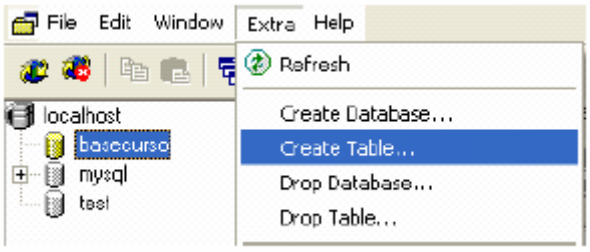
Para crear una base de datos nueva, hay que ir al menú ***Extra*** y la opción ***Create Database...***



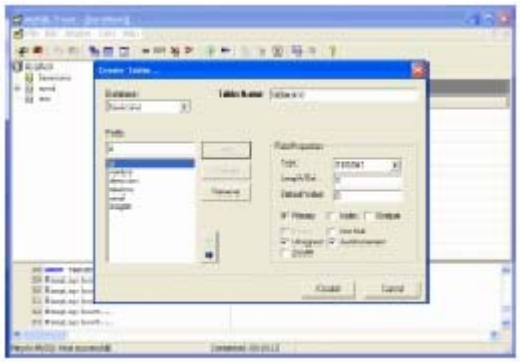
Hay que ingresar el nombre de la base de datos, podemos crear ya la base que utilizaremos en nuestros ejemplos de PHP, así que pongámosle de nombre ***basecurso***.



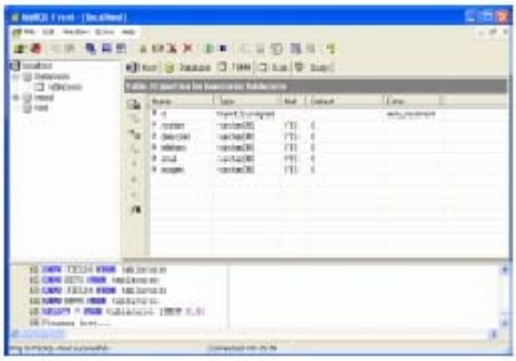
Ahora hay que crear una tabla donde almacenaremos los datos, asi que seleccionamos la nueva base y de nuevo vamos al menú ***Extra*** solo que ahora seleccionamos ***Create Table...***



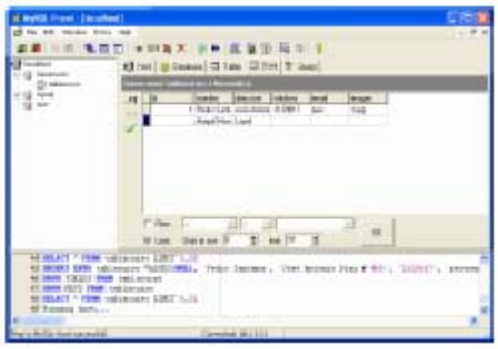
En la ventana que aparece ponemos el nombre de la tabla el cual será ***tablacurso***, en esa misma pantalla crearemos los campos, los cuales serán ***id*** (con propiedades de primario y auto incremento), ***nombre***, ***direccion***, ***telefono***, ***email*** e ***imagen*** (todos de tipo varchar), ya que los agregamos presionamos **Create!**



Nos mostrará la tabla creada con sus respectivos campos y propiedades.



También si nosotros lo deseamos podemos ingresar información a la base desde el mismo ***MySQL Front***, pero este no es el caso.



Ahora que si nosotros deseamos usar el método de línea de comandos del MySQL la
información de la tabla es esta:

```
CREATE TABLE tablacurso (
id tinyint(3) unsigned NOT NULL auto_increment,
nombre varchar(30) DEFAULT '0' ,
direccion varchar(30) DEFAULT '0' ,
telefono varchar(30) DEFAULT '0' ,
email varchar(30) DEFAULT '0' ,
imagen varchar(30) DEFAULT '0' ,
PRIMARY KEY (id)
```

);
Después de tanta información sobre MySQL ya es tiempo de iniciar a hacer programas
PHP para manejo de bases de datos de MySQL.

PHP PARA BASES DE DATOS MYSQL

Conectarse
Empecemos con el primer script, que nos mostrará como conectarnos a un base de datos
(*conectarse.php*).
conectarse.php

```
<html>
<head>
<title>Ejemplo de PHP</title>
</head>
<body>
<?php
function Conectarse()
{
if
(!($link=mysql_connect("localhost","pecesama","pruebas")))
{
echo "Error conectando a la base de datos.";
exit();
}
if (!mysql_select_db("basecurso",$link))
{
echo "Error seleccionando la base de datos.";
exit();
}
return $link;
}
Conectarse();
echo "Conexión con la base de datos conseguida.<br>";
?>
</body>
</html>
```

Como podemos ver en el ejemplo anterior aislé lo necesario para la conexión en una función, ahora esa función la pondremos en un archivo PHP solo (*conec.php*).

conec.php

```
function Conectarse()
{
if (!($link=mysql_connect("localhost","pecesama","pruebas")))
{
exit();
}
if (!mysql_select_db("basecurso",$link))
{
exit();
}
return $link;
}
```

Ya que tenemos la función en un archivo separado solo hay que mandarlo llamar
cuando sea necesario, de esta forma cuando tengamos aplicaciones complejas que use
muchas páginas php y sea necesario cambiarle algo a la conexión solo se le tenga que hacer
el cambio a este pequeño archivo y no a todas las páginas.

Agregar registros

Veremos un ejemplo de agregar registros a la base de datos (*insertareg.php* y *agregar.php*).

insertareg.php

```
<html>
<head>
<title>Ejemplo de PHP</title>
</head>
<body>
<H1>Ejemplo de uso de bases de datos con PHP y MySQL</H1>
<FORM ACTION="agregar.php">
<TABLE>
<TR>
<TD>Nombre:</TD>
<TD><INPUT TYPE="text" NAME="nombre" SIZE="20"
MAXLENGTH="30"></TD>
</TR>
<TR>
<TD>Direccion:</TD>
<TD><INPUT TYPE="text" NAME="direccion" SIZE="20"
MAXLENGTH="30"></TD>
</TR>
<TR>
<TD>Telefono:</TD>
<TD><INPUT TYPE="text" NAME="telefono" SIZE="20"
MAXLENGTH="30"></TD>
```

```
</TR>
<TR>
<TD>Email:</TD>
<TD><INPUT TYPE="text" NAME="email" SIZE="20"
MAXLENGTH="30"></TD>
</TR>
<TR>
<TD>Imagen:</TD>
<TD><INPUT TYPE="text" NAME="imagen" SIZE="20"
MAXLENGTH="30"></TD>
</TR>
</TABLE>
<INPUT TYPE="submit" NAME="accion" VALUE="Grabar">
</FORM>
<hr>
<?php
include("conec.php");
$link=Conectarse();

$result=mysql_query("select * from tablacurso",$link);
?>
<TABLE BORDER=1 CELLSPACING=1 CELLPADDING=1>
<TR>
<TD> Nombre</TD>
<TD> Dirección </TD>
<TD> Telefono </TD>
<TD> Email </TD>
<TD> Imagen </TD>
</TR>
<?php
while($row = mysql_fetch_array($result)) {
printf("<tr><td> %s</td><td> %s </td><td> 
;%s </td><td> %s </td><td> <img
src=%s> </td></tr>",
$row["nombre"],$row["direccion"],$row["telefono"],$row["email
"],$row["imagen"]);
}
mysql_free_result($result);
?>
</table>
</body>
</html>
```

agregar.php

```
<?php
include("conec.php");
```

```
$link=Conectarse();
$Sql="insert into tablacurso
(nombre,direccion,telefono,email,imagen) values
('$nombre','$direccion', '$telefono', '$email', '$imagen')";
mysql_query($Sql,$link);
header("Location: insertareg.php");
?>
```

Modificar registros

Veremos un ejemplo de modificar registros a la base de datos, consta de tres archivos diferentes, el primero para introducir la consulta por el campo nombre, el segundo
para realizar los cambios necesarios y el tercero para modificar la base de datos (*consulta.htm*, *busca.php* y *modifica.php*).

consulta.htm

```
<html>
<head>
<title>Ejemplo de PHP</title>
</head>
<body>
<H1>Ejemplo de modificar</H1>
<FORM ACTION="busca.php">
Nombre:
<INPUT TYPE="text" NAME="nombre" SIZE="20" MAXLENGTH="30">
<INPUT TYPE="submit" NAME="accion" VALUE="Buscar">
</FORM>
</body>
</html>
```

busca.php

```
<html>
<body>
<?php
include("conec.php");
$link=Conectarse();
$Sql="select * from tablacurso where nombre like '%$nombre%'";
echo $Sql;
$result=mysql_query($Sql,$link);
?>
<TABLE BORDER=1 CELLSPACING=1 CELLPADDING=1>
<TR>
<TD> Nombre</TD>
<TD> Dirección </TD>
<TD> Telefono </TD>
<TD> Email </TD>
<TD> Imagen </TD>
</TR>
```

```
<form name="form1" method="post" action="modifica.php">
<?php
while($row = mysql_fetch_array($result))
{
printf("<tr><td><INPUT TYPE='text' NAME='nombre' SIZE='20'
MAXLENGTH='30' value='%s'></td><td> <INPUT TYPE='text'
NAME='direccion' SIZE='20' MAXLENGTH='30'
value='%s'> </td><td> <INPUT TYPE='text'
NAME='telefono' SIZE='20' MAXLENGTH='30'
value='%s'> </td><td> <INPUT TYPE='text'
NAME='email' SIZE='20' MAXLENGTH='30'
value='%s'> </td><td> <INPUT TYPE='text'
NAME='imagen' SIZE='20' MAXLENGTH='30'
value='%s'> </td></tr>",
$row["nombre"],$row["direccion"],$row["telefono"],$row["email
"],$row["imagen"]);
}
mysql_free_result($result);
?>
</form>
</body>
</html>
```

modifica.php

```
<?php
include("conec.php");
$link=Conectarse();
$Sql="UPDATE tablacurso SET nombre='$nombre',
direccion='$direccion', email='$email', telefono='$telefono'
imagen='$imagen' WHERE nombre='$nombre'";
mysql_query($Sql,$link);
header("Location: consulta5.php");
?>
```

Eliminar registros

Pasemos a la eliminación de registros, este consta de dos archivos, los dos .php el primero es para elegir el registros a borrar y el segundo lo borra (*eliminareg.php* y *borra.php*).

eliminareg.php

```
<html>
<head>
<title>Ejemplo de PHP</title>
</head>
```

```
<body>
<H1>Ejemplo de uso de bases de datos con PHP y MySQL</H1>
<?php
include("conec.php");
$link=Conectarse();
$result=mysql_query("select * from tablacurso",$link);
?>
<TABLE BORDER=1 CELLSPACING=1 CELLPADDING=1>
<TR>
<TD> Nombre</TD>
<TD> Dirección </TD>
<TD> Telefono </TD>
<TD> Email </TD>
<TD> Imagen </TD>
<TD> Borra </TD>
</TR>
<?php
while($row = mysql_fetch_array($result)) {
printf("<tr><td> %s</td><td> %s </td><td> 
;%s </td><td> %s </td><td> %s </td><
/td><td><a href=\"borra.php?id=%d\">Borra</a></td></tr>",
$row["nombre"],$row["direccion"],$row["telefono"],$row["email
"],$row["imagen"],$row["ID"]);
}
mysql_free_result($result);
?>
</table>
</body>
</html>
```

borra.php

```
<?php
include("conec.php");
$link=Conectarse();
mysql_query("delete from tablacurso where ID = $id",$link);
header("Location: eliminareg.php");
?>
```