Programación Web
Utilizando Php y MySql

Introducción a la Programación Web

La Programación Web es la que se realiza sobre la plataforma de Internet, ya sea para construir páginas web dinámicas, aplicaciones para diversos tipos de transacciones de negocios para cierta empresa o portales para ofrecer varios servicios como foros, correo electrónico, noticias, etc.

Para realizar programación web se necesitará varias herramientas que conjuntamente conseguirán nuestros objetivos "*programar en Internet*". Necesitaremos conocimientos básicos acerca de HTML, JavaScript, diseño de páginas web y de las herramientas con la cual programaremos nuestras páginas dinámicas.

HTML

El HTML es el lenguaje con el que se definen las páginas web. Básicamente se trata de un conjunto de etiquetas que sirven para definir la forma en la que presentar el texto y otros elementos de una página.

El HTML se creó en principio con objetivos divulgatorios. No se pensó que la web llegara a ser un área de ocio con carácter multimedia, de modo que, el HTML se creó sin dar respuesta a todos los posibles usos que se le iba a dar y a todos los colectivos de gente que lo utilizaría en el futuro. Sin embargo, pese a esta deficiente planificación, si que se han incorporado modificaciones con el tiempo, estos son los estándares del HTML. Numerosos estándares se han presentado ya, el HTML 4.0.1 es el último estándar a febrero de 2001, sin embargo se puede encontrar información actualizada en www.w3.org, donde se publican estándares sobre todo lo relacionado con tecnologías para el world wide web (www).

El HTML es un lenguaje fácil de aprender, lo que permite que cualquier persona, aunque no haya programado en su vida pueda enfrentarse a la tarea de crear en web. HTML es fácil y pronto dominaremos este lenguaje. Más adelante se conseguirán resultados profesionales gracias a la capacidad de diseño y tu vena artística.

Una vez que conocemos el concepto del HTML, vamos a ver un poquito de cosas más. Este lenguaje se escribe en un documento de texto, por eso se necesita un editor de texto para escribir una página web. Así pues, el archivo donde está contenido el código HTML es un archivo de texto, con la peculiaridad, que tiene extensión .html o .htm (es indiferente cual de ellos usar). De modo que cuando programemos en HTML trabajaremos en un notepad por ejemplo y guardaremos los archivos como "mipagina.html".

El principio esencial del lenguaje HTML es el uso de las etiquetas (tags). Funcionan de la siguiente manera:

<XXX> → Este es el inicio de una etiqueta.
</XXX> → Este es el cierre de una etiqueta.

Las letras de la etiqueta pueden estar en mayúsculas o minúsculas, indiferentemente. Lo que haya entre ambas etiquetas estará influenciada por ellas. Por ejemplo, todo el documento HTML debe estar entre las etiquetas <HTML> y </HTML>:

<HTML> [Todo el documento] </HTML>

Un documento HTML en sí está dividido en dos zonas principales:

El encabezamiento, comprendido entre las etiquetas <HEAD> y </HEAD>
El cuerpo, comprendido entre las etiquetas <BODY> y </BODY>

Dentro del encabezamiento hay información del documento, que no se ve en la pantalla principal del BROWSER que es utilizado para visualizar el documento HTML, principalmente la información encontrada en el encabezamiento es el título del documento, comprendido entre las etiquetas <TITLE> y </TITLE>. El título debe ser breve y descriptivo de su contenido, pues será lo que vean los demás cuando añadan nuestra página a su bookmark (o agenda de direcciones).

Dentro del cuerpo está todo lo que queremos que aparezca en la pantalla principal (texto, imágenes, etc.) Por tanto, la estructura de un documento HTML queda de esta manera:

```
<HTML>
<HEAD>
<TITLE> Título de la página </TITLE>
</HEAD>
<BODY>
[Aquí van las etiquetas que visualizan la página]
</BODY>
</HTML>
```

Más adelante se verá más acerca de las etiquetas y como funcionan, de acuerdo se vaya introduciendo en la construcción de páginas web utilizando tecnología php.

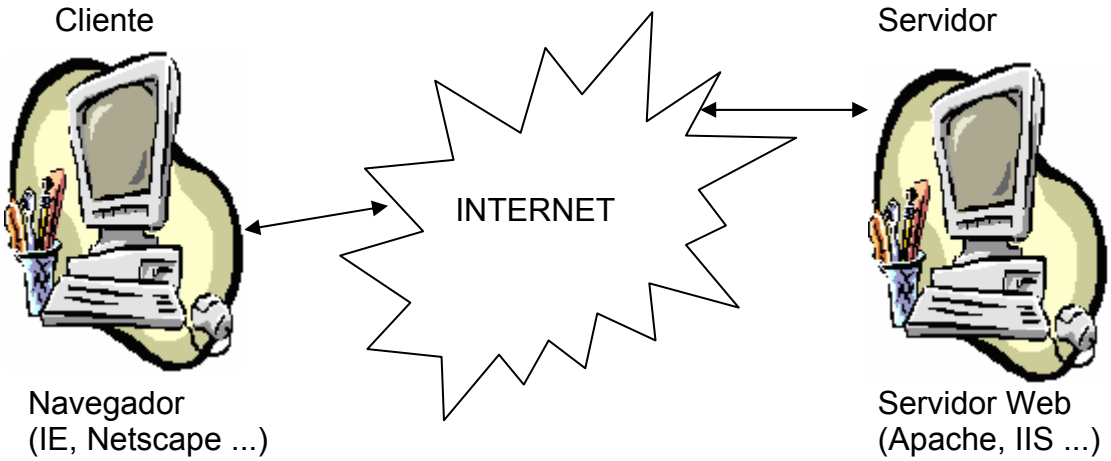
Servidores Web

Un servidor web es un software que se instala sobre una máquina que va a funcionar como servidor en la red de Internet y en esta se van a almacenar páginas web (páginas HTML o páginas php). Cabe resaltar que el servidor es una computadora (hardware) y el servidor web es un programa (software).

El servidor web está siempre esperando peticiones de algún usuario que desde algún lugar en el Internet le hace solicitudes de alguna página y este le responde con la información deseada. El usuario hace la petición a través de la URL *Uniform Resource Location* (referencia a un recurso en Internet) por medio de un navegador, la cual indica la dirección donde se aloja la página exacta que se desea. Ej:

http://www.php.net/downloads.php
|_dirección_____| página solicitada |

Este petición lo hace desde un navegador para Internet (browser) el cual es un programa instalado en la máquina del cliente y esta petición viaja a través de Internet.



Para efectos de desarrollo en este curso se utilizará una sola máquina que simule este proceso, usando un servidor web que almacena sus páginas en un directorio de la máquina (directorio donde se instaló el programa) y el navegador que este instalado en la misma máquina local para acceder a las páginas deseadas.

Las principales características de configuración de un servidor web son:

El puerto. El cual tiene que ser único perteneciente a este software para que no exista conflictos con otros programas que escuchen peticiones por el mismo puerto.
Ej. Port 8080

El Directorio raíz. Es el directorio / donde las páginas se van a almacenar similar a una estructura de ficheros de un sistema operativo, en el cual se puede almacenar carpetas y subcarpetas que contengan archivos (páginas). Este directorio será el indicado por la configuración del programa.
Ej.: DocumentRoot C:\Archivos de Programa\Apache\htdocs

El Nombre del Servidor. Indica el servidor donde se alojan las páginas, donde se ha instalado el servidor web. Este nombre podrá ser un dominio como www.apache.org, www.goole.com o en nuestro caso por ser una máquina de desarrollo se escribirá localhost.
Ej.: ServerName localhost

Estas configuraciones se realizan en un archivo que sirva para este fin perteneciente al servidor web. En el caso de apache se llama httpd.conf (ver anexo)

PHP

PHP -acrónimo de « Hypertext Preprocessor»- suele definirse como: un lenguaje “Open Source” *interpretado*, de *alto nivel*, cuyo código va *insertado (embebido)* en páginas HTML y que es *ejecutado en el servidor*. Es un lenguaje de estilo clásico, cercano en su sintaxis a JavaScript o a C++.

La gran diferencia con los otros lenguajes -Java o JavaScript- es que mientras que estos se ejecutan en el navegador, PHP se ejecuta en el servidor y envía los resultados al navegador en forma de página web. Un ejemplo:

```
<html>
    <head>
        <title>Example</title>
    </head>
    <body>
        <?
        echo "Mi primer Script";
        ?>
    </body>
</html>
```

Hemos escrito código HTML con cierto código PHP embebido (introducido) en el mismo, que producirá cierta salida (en nuestro ejemplo, producirá un texto). El código PHP se incluye entre etiquetas especiales de comienzo y final <? ?> que nos permitirán entrar y salir del modo PHP.

Con este script alojado en el servidor web, el cliente solamente recibirá el resultado de la ejecución del lado del servidor, sin ninguna posibilidad de determinar que código ha producido el resultado recibido. El servidor web puede ser incluso configurado para que procese todos los ficheros HTML con PHP.

La instrucción *echo* seguida de una cadena de texto entrecomillada hará que el PHP escriba en la página web resultante lo contenido en esa cadena de texto. Puedo insertar dentro de la cadena de texto que sigue a *echo* cuantas etiquetas HTML desee. Podría incluso escribir el código completo una página web.

Sintaxis PHP

Como ya hemos visto tenemos las etiquetas con las cuales se reconocerá el código php. Veremos para comenzar que las líneas de código se separan con el símbolo de “;” (punto y coma) al igual que el lenguaje C.

Comentarios

Hay 2 formas:

- Para una línea basta colocar // al comienzo de la misma. También se puede usar el símbolo # .
- Para varias líneas Se coloca /* al comienzo de la primera línea de comentario y */ al final de la última. Los comentarios de varias líneas no pueden anidarse. Si por error los anidamos, PHP nos dará un mensaje de error.

Constantes

En PHP las constantes se definen de la siguiente forma:

```
define("Nombre","Valor")
```

Los valores asignados a las constantes se mantienen, incluso cuando tales constantes son invocadas desde una función. No es necesario escribir entre comillas los valores de las constantes cuando son numéricas.

```
<?
/* Definiremos la constante Pesos y le asignamos el valor 2600.25*/
define(Pesos",2600.25);

# Definimos la constante Cadena
define("Cadena","Esta constante es una cadena");

// comprobamos los valores
echo "Valor de la constante Pesos: ", Pesos, "<BR>";
echo "Valor de la constante Cadena: ", Cadena, "<BR>";
?>
```

Con la instrucción *echo* pueden enlazarse cadenas de caracteres, constantes y variables sin más que separarlas con una *coma*.

Existen constantes predefinidas por el lenguaje algunas de ellas son:

__FILE__ Recoge el nombre del fichero que se está ejecutando y la ruta completa de su ubicación en el servidor.

__LINE__ Recoge el número de línea (incluidas líneas en blanco) del fichero PHP que se está interpretando

PHP_OS Recoge información sobre el Sistema Operativo que utiliza el servidor en el que se está interpretando el fichero

PHP_VERSIÓN Recoge la versión de PHP que está siendo utilizada por el servidor.

Variables

Todas los nombres de variables en PHP tienen que empezar por el símbolo $ y se asignan contenido con el símbolo =. El nombre de las variables debe empezar por una letra. $a1 es un nombre válido, pero $1a no es un nombre válido

PHP diferencia entre mayúsculas y minúsculas. $pepe es una variable distinta de $Pepe.

En PHP no es necesario definir el tipo de variable, por lo tanto una misma variable puede contener en un momento determinado una cadena de caracteres y posteriormente, la misma variable puede contener un valor numérico, susceptible de ser operado matemáticamente.

Si definimos dentro de una función una variable con idéntico nombre que otra ya definida fuera de ella podemos asignarle un valor distinto que será utilizado únicamente dentro del ámbito de la función. Para poder utilizar el valor de una variable -previamente definida- dentro de una función es preciso escribir dentro de la función *global nombre de la variable*

Por ejemplo: global $a1;

```
<?
# Defino la variable $pepe como vacía
$pepe="";

# Defino la variable $Pepe y $Pepa (ojo con mayúsculas y minúsculas)
$Pepe="Me llamo Pepe y soy serio y formal";
$Pepa="Me llamo Pepa y también soy seria y formal";

echo "<br> El valor de la variable pepe es: ",$pepe;
echo "<br> El valor de la variable Pepe es: ",$Pepe;

# defino una función para invocar variable con mismo nombre
function vervariable(){
      echo "<br> Si invoco la variable Pepe desde una función";
      echo "<br>me aparecerá en blanco";
      echo "<br>El valor de la variable Pepe es: ",$Pepe;
}

#llamo a la función
vervariable();

# defino una función para invocar variable con mismo nombre pero con global
function ahorasi(){
      global $Pepe;

      echo"<br><br> Como he puesto en la función, el ámbito global";
      echo "<br>ahora Pepe aparecerá";
      echo "<br>El valor de la variable Pepe es: ",$Pepe;
}
#llamo a la función
ahorasi();
?>
```

Fíjate cono se definen las funciones:

```
function nombre (){
... instrucciones ...
}
```

Las instrucciones que contiene una función no se ejecutan mientras que no se invoque esa función -desde una etiqueta PHP- con esta sintaxis:

```
nombre();
```

Existen también variables predefinidas en php las cuales pueden facilitar información del servidor donde está instalado y del cliente que esta accediendo al servidor, conforme se vayan necesitando se irán utilizando las mismas. Hay que tener en cuenta que el valor de estas variables dependerán del servidor donde este instalado. Como ejemplo realizaremos un script que dará como resultado todas las variables y su respectivo valor del servidor con el cual estas programando, este script se llamará “info.php”

El script solo contiene la siguiente instrucción:

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

Como ya mencionamos en PHP no es necesaria una definición previa de tipo de variables, dependiendo de los valores que se les vayan asignando las variables

cambian de tipo y se adaptan a los nuevos valores. Las variables en PHP pueden ser de tres tipos:

- Enteras (tipo *Integer*)
- De coma flotante (tipo *Double*)
- Cadenas (tipo *String*)

Cuando asignamos a una variable un valor numérico comprendido entre - 2^{31} y + 2^{31} PHP la interpreta como tipo *Integer*. Si el valor numérico de una variable *desborda* el intervalo anterior -bien por asignación directa o como resultado de una operación aritmética- PHP convierte la variable automáticamente en tipo *Double*.

Cualquier variable a la que se le asigne como valor una cadena de caracteres (números y/o letras encerrados por comillas) es interpretada por PHP como tipo *String*.

Para obtener el tipo de una variable utilizaremos la función gettype(*variable*), la cual devuelve una cadena de caracteres con el tipo de la variable que contiene. Los valores que puede devolver son: *Integer*, *double* o *string.*

PHP permite forzar los tipos de variables. Eso quiere decir que se puede obligar a PHP a que asigne un determinado tipo a una determinada variable.

Para forzar una variable a tipo *double* basta con anteponer a su valor una de estas expresiones (entre paréntesis): (*double*), o (*real*) o (*float*).

Por ejemplo para convertir la variable $a a tipo *Double*.

```
$a=((double)45);        ó
$a=((float)45);         ó
$a=((real)45);
```

Para forzar una variable a tipo *Integer* basta con *anteponer* a su valor una de estas expresiones (entre paréntesis): (*integer*), o (*int*).

Por ejemplo para convertir la variable $b a tipo *Integer*.:

```
$b=((integer)4.5);      ó
$b=((int)45);
```

Para forzar una variable a tipo *String* basta con anteponer a su valor (entre paréntesis): *(string).*

Por ejemplo para convertir la variable $c a tipo *String*.

```
$c=((string)4.5);
```

Operadores

Los operadores en php se dividen en:

- Aritméticos:

$a + $b Suma

$a - $b	Resta
$a * $b	Multiplicación
$a / $b	División
$a % $b	Resto de la división de $a por $b
$a++	Incrementa en 1 a $a
$a--	Resta 1 a $a
Sqrt($a)	Raíz cuadrada
pow($a,$b)	Potencia $a elevada a la $b.
log($a)	Logaritmo natural de $a
Log10($a)	Logaritmo base 10 de $a
Exp($a)	Exponencial de a

Las funciones trigonométricas utilizan radianes como parámetros

Sin($a)	Seno de $a
Cos($a)	Coseno de $a
Tan($a)	Tangente de $a

- Cadenas:

El único operador de cadenas que existen es el de concatenación, el punto. Pero PHP dispone de toda una batería de funciones que permitirán trabajar cómodamente con las cadenas.

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

- Comparación:

Aquí hay que señalar que PHP identifica como verdadero a 1 y 0 a falso.

$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

- Lógicos:

$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, y recíprocamente

- 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

Cadenas

A las variables tipo *string* pueden asignársele valores de dos formas:

- Escribiendo el contenido entre comillas después de signo igual.

```
$cadena="Texto del contenido";
```

- Utilizando la sintaxis de documento incrustado

```
$cadena= <<<EOD
... contenido ...
... puede ir ....
.. en varias líneas...
EOD;
```

EOD representa una palabra cualquiera que debe repetirse exactamente igual al final de la instrucción.

Recordemos también que el único operador para concatenar cadenas es el . (punto).

```
<?
$cadena1="Esto es una cadena de texto";
$cadena2="Esta es la segunda cadena";
echo $cadena1.$cadena2,"<br>";
$cadena3=$cadena1.$cadena2;
$cadena3 .=" Este es el texto que se añadirá a la variable cadena3";
echo $cadena3,"<br>";
$cadena3 .=" Otro añadido más";
echo $cadena3,"<br>";
$cadena3 .= <<<Pepito
 Ahora le añado a la cadena
este trocillo asignado con el formato
de documento incrustado
Pepito;
echo $cadena3,"<br>";
?>
```

Existen muchas funciones para manejar cadenas, a continuación nombraremos algunas:

- chr(*n*): Devuelve el carácter cuyo código ASCII es n
- strlen(*cadena*): Devuelve la longitud (número de caracteres, incluso espacios) de la cadena
- strtolower(*cadena*): Cambia los caracteres de la cadena a minúsculas
- strtoupper(*cadena*): Convierte en mayúsculas todos los caracteres de la cadena
- ltrim(*cadena*): Elimina los espacios al principio de la cadena
- rtrim(*cadena*): Elimina los espacios al final de la cadena
- trim(*cadena*): Elimina los espacios tanto al principio como al final de la cadena

- substr(*cadena*,n): Si el valor de n es positivo extrae todos los caracteres de la cadena a partir del que ocupa la posición *enésima* a partir de la izquierda. Si n es negativo extrae los *n* últimos caracteres de la cadena
- substr(*cadena*,n,m): Si n y m son positivos extrae m caracteres a partir del que ocupa la posición enésima contados de izquierda a derecha. Si n es negativo y m es positivo extrae m (contados de izquierda a derecha) a partir del que ocupa la posición enésima contada de derecha a izquierda. Si n es positivo y m es negativo extrae la cadena comprendida entre el enésimo carácter (contados de izquierda a derecha) hasta el emésimo partir contado de derecha a izquierda. Si n es negativo y m también es negativo extrae la cadena comprendida entre el emésimo y el enésimo caracteres contado de derecha a izquierda. Si el valor absoluto de n es menor que el de m devuelve una cadena vacia.
- strrev(*cadena*):Devuelve la cadena invertida
- strcmp($A,$B): Compara ambas cadenas discriminando entre mayúsculas y minúsculas. Cuando son idénticas devuelve cero, en caso de no ser iguales devuelve +1 ó -1.
- strncmp($A,$B,n): Similar a strcmp(). Discrimina mayúsculas de minúsculas y solo compara los n primeros caracteres de ambas cadenas

Arreglos

Un arreglo (array) es sencillamente una tabla de valores. Cada uno de ellos se identifica y se asigna mediante una variable ($*nombre*) seguida de un corchete ([ ]) que contiene el índice del array.

El índice puede ser escalar (equivaldría al número de fila de la tabla, por poner un ejemplo) o puede ser asociativo que equivaldría en alguna medida al nombre de la fila de la tal tabla. En el caso de los índices escalares, estos comienzan desde CERO.

Arrays escalares

Los elementos de un *array* escalar puede escribirse usando la siguiente sintaxis:

```
$a[]=valor  ó
$a[xx]=valor
```

En el primero de los casos, PHP asigna automáticamente como índice el valor siguiente al último asignado. Si es el primero que se define, le pondrá como índice 0 (CERO). En el segundo de los casos, seremos nosotros quienes pongamos (xx) el número correspondiente al valor del índice. Si al índice ya se le hubiera asignado un valor, cambiará el valor de la variable, en caso contrario creará un nuevo elemento del *array*.

Arrays asociativos

Los elementos de un *array* asociativo puede escribirse usando la siguiente sintaxis:

```
$a["índice"]=valor
```

En este caso, el índice será una cadena y se escribirá entre comillas.

```
<?
# Crearé dos arrays escalares, $a y $b
$a[0]="Domingo"; $a[1]="Lunes";

# si pongo corchetes vacíos va añadiendo índices automáticamente

$a[]="Martes"; #equivale a escribir $a[2]
$a[]="Miércoles"; #equivale a escribir $a[3]
$a[]="Jueves"; #equivale a escribir $a[4]
$a[]="Viernes";
$a[]="Sábado";
echo "Al imprimir $a[1] escribirá el 2° elemento: ",$a[1],"<br>";

# puedo iniciar otro array sin indicar índices
# PHP empezara a contar a partir de CERO
$b[]="Domingo";
$b[]="Lunes";
$b[]="Martes";$b[]="Miércoles";$b[]="Jueves";
$b[]="Viernes";$b[]="Sábado";
echo "Al imprimir $b[4] escribirá el 5° elemento: ",$b[4],"<br>";

# ahora un array asociativo (un nombre y un valor)
$c["Primero"]="Domingo";$c["Segundo"]="Lunes";$c["Tercero"]="Martes";
$c["Cuarto"]="Miércoles";$c["Quinto"]="Jueves";$c["Sexto"]="Viernes";
$c["Séptimo"]="Sábado";
echo "Al imprimir $c["Tercero"] escribirá: ",$c["Tercero"],"<br>";
?>
```

Un array bidimensional recoge valores de una tabla de doble entrada. Cada uno de los elementos se identifica y se asigna mediante una variable ($*nombre*) seguida de dos ([]) que contienen los índices del array.

Los índices pueden ser escalares (equivaldrían al número de fila y columna que la celda ocuparía en la tabla), o puede ser asociativo que equivaldría en alguna medida a usar como índices los nombres de la fila y de la columna.

Siguen las mismas reglas que los unidimensionales al momento de asignar valores, ya sea automáticamente (secuencialmente) o en un índice predeterminado:

```
$a[][]=valor      ó
$a[xx][]=valor    ó también
$a[][xx]=valor    ó
$a[xx][yy]=valor
```

En el caso de un *array* bidimensional asociativo:

```
$a["indice1"]["indice2"]=valor
```

PHP permite el uso de arrays de dimensiones superiores a dos. Bastaría con añadir corchetes para modificar la dimensión del array.

Para asignar valores a una matriz puede usarse la función array() que tiene la siguiente sintaxis:

```
$a= array (
indice 0 => valor,
..... ,
indice n => valor,
```

```
);
```

Por ejemplo:

```
$z=array (
0 => 2,
1 => "Pepe",
2 => 34.7,
);
```

produciría el mismo resultado que escribir:

```
$z[0]=2;
$z[1]="Pepe";
$z[2]=34.7;
```

a continuación un ejemplo con arreglos bidimensionales, utilizando etiquetas HTML que permiten formar tablas.

```
<html>
<head>
<title>Resultados de Partidos</title>
</head>
<body>
<?
# resultado de los encuentros en los que el Liga  local
$b["Liga"]["Liga"]="-";$b["Liga"]["Quito"]="3-2";
$b["Liga"]["Barcelona"]="5-3";

# resultado de los encuentros en los que el Quito es local
$b["Quito"]["Liga"]="0-11";$b["Quito"]["Quito"]="-";
$b["Quito"]["Barcelona"]="2-1";

# resultado de los encuentros en los que el Barcelona es local
$b["Barcelona"]["Liga"]="0-0";$b["Barcelona"]["Quito"]="1-3";
$b["Barcelona"]["Barcelona"]="-";
?>
<font face="Arial">
<table width="75%" border="1" name="tabla" align="center">
  <tr align="center">
    <td colspan="4">Resultados</td>
  </tr>
  <tr>
    <td>Indice</td>
    <td>Liga</td>
    <td>Quito</td>
    <td>Barcelona</td>
  </tr>
  <tr>
    <td>Liga</td>
    <td><?echo $b["Liga"]["Liga"];?></td>
    <td><?echo $b["Liga"]["Quito"];?></td>
    <td><?echo $b["Liga"]["Barcelona"];?></td>
  </tr>
  <tr>
    <td>Quito</td>
    <td><?echo $b["Quito"]["Liga"];?></td>
    <td><?echo $b["Quito"]["Quito"];?></td>
    <td><?echo $b["Quito"]["Barcelona"];?></td>
  </tr>
  <tr>
    <td>Barcelona</td>
```

```
    <td><?=$b["Barcelona"]["Liga"]?></td>
    <td><?= $b["Barcelona"]["Quito"] ?></td>
    <td><?=$b["Barcelona"]["Barcelona"];?></td>
  </tr>
</table>
</font>
</body>
</html>
```

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
}
```

En algunos casos resulta útil y cómodo el uso del condicional ternario. Su sintaxis es esta:

```
(condición) ? (opc1) : (opc2)
```

Si se cumple la condición se ejecuta la opc1 pero en el caso de que no se cumpla se ejecutará la opc2. Este condicional está limitado prácticamente a la asignación de valores a variables.

```
<? $a=5;
($a==8) ? ($B="El valor de a es 8") : ($B="El valor de a no es 8");
echo $B;
?>
```

switch....case.....default

Es una alternativa a if...*else*, la sentencia *switch* evalúa y compara cada expresión de la sentencia *case* con la expresión que evaluamos, si llegamos al final de la lista de *case* y no 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*.

```
<?
switch ($dia) {
      case "Lunes":
            echo "Hoy es Lunes";
            break;
      case "Martes":
```

```
            echo "Hoy es Martes";
            break;
      case "Miercoles":
            echo "Hoy es Miércoles";
            break;
      case "Jueves":
            echo "Hoy es Jueves";
            break;
      case "Viernes":
            echo "Hoy es Viernes";
            break;
      default:
            echo "Esa cadena no es ningún día de la semana";
}
?>
```

while

La sentencia while (condición) repite las instrucciones que le acompañan en tanto y cuanto se cumpla la condición. La sintaxis es esta:

```
while(condición){
      ...instrucciones ...
}
```

Podemos romper un bucle WHILE utilizando la sentencia BREAK.

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

do while

La sentencia do... while (condición) es muy similar -en cuanto a su aplicación- a while. La única diferencia es que do ....while comprueba la condición después de ejecutar las instrucciones, mientras que while lo hace antes, por lo que el bloque de código se ejecuta siempre al menos una vez.

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

for

```
for ( desde ; hasta ; incre ){
...instrucciones....
}
```

Una de las sintaxis del bucle for tiene características muy similares a las de esta misma instrucción en otros lenguajes de programación. El parámetro *desde* permite asignar un valor inicial a una variable contador de iteraciones. El parámetro *hasta* permite establecer el valor final del contador de iteraciones y con *incre* se pueden establecer los incrementos o decrementos de la variable contador en cada iteración del bucle. Las instrucciones contenidas entre { } serán ejecutadas cada vez que se reitere el bucle.

El bucle for también se puede romper mediante la sentencia *break*.

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

foreach

La función foreach sólo es aplicable a un array tanto escalar como asociativo. Con un array (previamente definido) esta función va recogiendo en una nueva variable (var) los sucesivos valores de cada uno de los elementos del array. Las instrucciones escritas dentro de las { } permiten la visualización u operación con los sucesivos valores de los elementos del array.

```
foreach ( array as var ){ }
```

```
<?
$a=array("a","b","c","d","e");
$b=array(
 "uno"  =>"Primer valor",
 "dos"  =>"Segundo valor",
 "tres" =>"Tecer valor",
);

foreach($a as $pepe) {
echo $pepe,"<br>";
}
?>
```

La función continue

Si la función *break* permite interrumpir el desarrollo de un bucle y salir de él, la función *continue* permite interrumpir la ejecución de las instrucciones que le siguen y continuar en el paso siguiente iteración del mismo bucle.

```
<?
for ($i=0;$i<=10;$i++){
      #condicion de  multiplo de 2
      if ($i % 2 ==0 ) {
            continue; #Continúa con la siguiente iteración del for
      }
      echo "La variable I vale ",$i,"<br>";
}
?>
```

Funciones

Las funciones PHP definidas por el usuario tienen la siguiente sintaxis:

```
function nombre(){
.....
...instrucciones...
.....
}
```

Es imprescindible respetar estrictamente la sintaxis: function, () y escribir dentro de las llaves el código correspondiente a esa función.

Las funciones PHP no se ejecutan en tanto y cuanto no sean *invocadas*. Para invocar una función la sintaxis es esta:

```
nombre();
```

Al ser llamada con esta sintaxis, se ejecutan las instrucciones contenidas en la función. Aunque el versiones anteriores de PHP era necesario definir la función antes de invocarla, a partir de la versión 4 no es necesaria esa organización secuencial. La función y la llamada pueden estar escritas en cualquier parte del documento.

Utilizando valores externos

Si se invoca una variable definida previamente desde una función tomará valor NULO. Recuerda que cuando estudiamos las variables PHP veíamos que se excluían las funciones del ámbito de las variables, salvo que, se asignara a las mismas ámbito global dentro de la propia función.

Una de las formas de asignar valores a las variables internas es definir variables y valores dentro del paréntesis que va detrás de function nombre, así por ejemplo:

```
function p1($a=2,$b=5)
```

Esos valores 2 y 5 serían utilizado por la función independientemente de que las variables $a y/o $b pudieran contener otros distintos en el ámbito externo de la función.

Otra forma de asignación de valores a las variables internas de una función es pasar los *parámetros por valor*. En la definición de la función deben incluirse tantas variables como valores distintos pretendan pasarse. Para hacerlo la sintaxis es la siguiente:

```
function p1($a,$b)
```

Prepararía la función para recibir dos valores pero requeriría que además, en la llamada a la función se escribieran esos dos valores separados por comas.

```
p1(47,-32);
```

Una variante del ejemplo anterior, sería sustituir -en la llamada a la función- las constantes por variables predefinidas.

Pasando parámetros por referencia

Tal como estamos viendo, por defecto las funciones PHP pueden recibir valores de variables externas y utilizar esos valores sin que el valor original de la variable sufra modificación.

Una manera de forzar que la función modifique los valores externos de la variable es lo que se llama en argot «pasar por referencia». Para hacerlo hay que anteponer al nombre de la variable el símbolo & y PHP interpretará que la estamos pasando por referencia. El & puede anteponerse tanto en la definición de la función como en la llamada a la función.

```
function p1(&$a,&$b)
p1(&$a,&$b);
```

La segunda de las opciones nos concede mayor libertad ya que nos permite usar una sola función y decidir en las llamadas la forma de pasar los parámetros, con lo cual una misma función puede ser llamada tanto por valores como por referencia.

(para realizar esto último es necesario que la variable allow_call_time_pass_reference en el fichero php.ini este activada en On.)

```
<? $a=5; $b=47;
function a1(){
global $a;
echo "Resultado de 2*\$a en la función a2: ", 2*$a,"<br>";
echo "el valor de \$a dentro de la función es: ",$a;
}
a1();
echo "El valor de \$a después de la función es: ",$a,"<br><br>";

function a2($a=56){
echo "Resultado de 2*\$a en la función a3: ", 2*$a,"<br>";
echo "el valor de \$a dentro de la función a3 es: ",$a,"<br>";
}
a2();
echo "El valor de \$a despues de la función es: ",$a,"<br><br>";

function a3($a,$b){
echo "Resultado de 2*\$a+3*\$b en la función a4: ", 2*$a+3*$b,"<br>";
echo "el valor de \$a dentro de la función a4 es: ",$a,"<br>";
}
a3(7,-2);
echo "El valor de \$a despues de la función es: ",$a,"<br><br>";

function a4($a,$b){
echo "Resultado de 2*\$a+3*\$b en la función a5: ", 2*$a+3*$b,"";
echo "el valor de \$a dentro de la función a5 es: ",$a,"";
}
a4($a,$b);
echo "El valor de \$a despues de la función es: ",$a,"<Br><BR>";

function a5(&$a,$b){
$b=pow($b,2);
$a=pow($a,2);
echo "El nuevo valor de a dentro de la función es: ",$a, "<br>";
echo "El nuevo valor de b dentro de la función es: ",$b, "<br><br>";
}
```

```
a5($a,$b);
echo "Al salir de la función a vale: ",$a, "<br>";
echo "Al salir de la función b vale: ",$b, "<br>";

function a6($a,$b){
$b=pow($b,1/2);
$a=pow($a,2);
echo "El nuevo valor de a dentro de la función es: ",$a, "<br>";
echo "El nuevo valor de b dentro de la función es: ",$b, "<br><br>";
}
a6($a,&$b);
echo "Al salir de la función a vale: ",$a;
echo "Al salir de la función b vale: ",$b;
?>
```

Php puede recoger valores de variables mediante la instrucción *return* y asignar a cualquier variable que tomaría el tipo del valor devuelto, este puede ser entero, cadena, double, arreglo.

```
function a1($a,$b){
    return pow($a,$b);
}

function a2($a,$b){
    for ($i=0;$i<=$b;$i++){
         $z[]=pow($a,$i);
    }
    return $z;
}
```

Envío de Formularios

PHP permite enviar datos desde un formulario hasta un script PHP que pasan a estar disponibles en este como variables. Se requieren dos páginas, una para enviar y otra para recoger los datos enviados.

Para realizar esa transferencia es necesario escribir en HTML un formulario y poner dentro la etiqueta <form> los parámetros: action="nombre de la página que recibe los datos" y el método que por el momento puede ser get o post que se escribirían así: method="post" o method="get"

Al pulsar en el botoncito “Enviar”, el browser automáticamente transfiere los valores contenidos en cada campo del formulario y que son recogidos en variables con idéntico nombre al que figura en la opción name de cada campo del formulario. Esto es válido cuando en el fichero php.ini la variable Register globals está activada (=On).Sin embargo desde versiones actuales es recomendable usar esta variable desactivada por razones de seguridad y en ese caso las variables se reciben de la siguiente manera:

Si se enviaron con method="post": $HTTP_POST_VARS[‘name’]

Si se enviaron con method="get": $HTTP_GET_VARS[‘name’]

Hay que tomar en cuenta las comillas y que el nombre de las variables son sensibles a mayúsculas y minúsculas.

En versiones desde PHP 4.2.0 o superior, se puede utilizar las nuevas variables predefinidas:

$_ENV, $_POST y $_GET

que tienen la misma funcionalidad que las respectivas:

$HTTP_ENV_VARS, $HTTP_POST_VARS, y $HTTP_GET_VARS

Un ejemplo: Creamos un archivo con código html, lo llamaremos formu.php

```
<HTML>
<HEAD>
<TITLE>Ejemplo</TITLE>
</HEAD>
<BODY>
<form action='resultado.php' method='post'>
Escribe tu nombre:
<input type='text' name='nombre' value='' size=15><br>
Escribe tu clave:
 <input type='password' name='clave' value=''><br>

Elige tu color de coche favorito:<br>
<input type='radio' name='color' value='Rojo'>Rojo</br>
<input type='radio' name='color' value='Verde'>Verde</br>
<input type='radio' name='color' value='Azul'>Azul</br>

Elige los extras:<br>
<input type='checkbox' name="acondicionado" value="Aire">
Aire acondicionado<br>
<input type='checkbox' name="tapiceria" value="Tapicieria">
Tapiceria en piel<br>
<input type='checkbox' name="llantas" value="aluminio">
Llantas de aluminio<br>

¿Cual es el precio máximo que estarías dispuesto a pagar?<br>
<select name="precio">
<Option>Menos de 6.000 euros</option>
<Option>6.001 - 8.000 euros</option>
<Option>8.001 - 10.000 euros</option>
<Option>10.001 - 12.000 euros</option>
<Option>12.001 - 14.000 euros</option>
<Option>Más de 14.000 euros</option>
</select>

<textarea rows=5 cols=50 name='texto'></textarea>

<input type="submit" value="enviar">
<input type="reset" value="borrar">
</BODY>
</HTML>
```

Luego creamos el archivo resultado.php que tomaría la los datos del formulario. Si está la variable register_globals=On.

```
<HTML>
<HEAD>
<TITLE>Resultado</TITLE>
</HEAD>
<BODY>
```

```
<?
echo "El method que ha usado fué: ",$REQUEST_METHOD,"<br>";

echo $nombre,"<br>";
echo $clave,"<br>";
echo $color,"<br>";
echo $acondicionado,"<br>";
echo $tapiceria,"<br>";
echo $llantas,"<br>";
echo $precio,"<br>";
echo $texto,"<br>";
?>
</BODY>
</HTML>
```

Y en caso de tener register_globals = Off el archivo sería

```
<HTML>
<HEAD>
<TITLE>Resultado</TITLE>
</HEAD>
<BODY>
<?
echo "El método usado fué: ",$HTTP_ENV_VARS[REQUEST_METHOD],"<br>";
echo  $HTTP_POST_VARS['nombre'],"<br>";
echo  $HTTP_POST_VARS['clave'],"<br>";
echo  $HTTP_POST_VARS['color'],"<br>";
echo  $HTTP_POST_VARS['acondicionado'],"<br>";
echo  $HTTP_POST_VARS['tapiceria'],"<br>";
echo  $HTTP_POST_VARS['llantas'],"<br>";
echo  $HTTP_POST_VARS['precio'],"<br>";
echo  $HTTP_POST_VARS['texto'],"<br>";
?>
</BODY>
</HTML>
```

MySql con PHP

Una de las grandes posibilidades de PHP es la posibilidad de manejar bases de datos alojadas en ordenadores remotos. Existen multitud de paquetes de gestión de bases de datos y PHP dispone de funciones para el manejo del algunos de ellos tales como: dBase, DBM, Microsoft SQ, PostgreSQL, mSQL, InterBase, MySQL. En este curso utilizaremos MySql por su versatilidad a la hora de interactuar con php, ya que este tiene funciones que manejan código a bajo nivel, donde permite una comunicación de alta velocidad.

Conexión con el servidor de bases de datos

Antes de empezar a trabajar con la base de datos debes cerciorarte de que tanto tu servidor Apache como tu WinMySQLAdmin están conectados (ver anexo).

La conexión con una base de datos remota tiene la siguiente sintaxis:

```
$link=mysql_connect (host, user, password)
```

donde $c es la variable que recoge el indentificador del enlace, host es la dirección del servidor de bases de datos, (en nuestra configuración local sería localhost) ,

user sería el nombre de uno de los usuarios registrados en la BD y password la clave de acceso correspondiente al user.

Para cerrar la conexión, PHP dispone de la siguiente función:

```
$link=mysql_close ($link)
```

donde $link es el nombre de la variable en la que se recogió el indentificador del enlace en el momento de la apertura.

```
<?
if($conexion=mysql_connect ("localhost","root","clave")){
      echo "<h2> Conexión establecida con el servidor</h2><br>";
      echo "El identificador de conexion es:",$conexion;
      if(mysql_close($conexion)){
            echo "<h2> Conexión cerrada con exito</h2><br>";
            echo "El identificador de conexion es:",$conexion;
      }else{
            echo "<h2> No se ha cerrado la conexión</h2>";
      }
}else{
      echo "<h2> NO HA SIDO POSIBLE ESTABLECER LA CONEXIÓN</h2>";
}
?>
```

Si realizáramos una segunda conexión (con los mismos argumentos) sin haber cerrado la anterior no se efectúa un nuevo enlace sino que nos devolverá el ya abierto.

Interacción Php con MySql

Php ofrece una gama de funciones para interactuar con MySql. Tiene funciones tanto para crear y borrar Bases de Datos, puede consultar los nombres da las tablas, campos, bases de datos disponibles en MySql y obtener información de la instalación del Servidor de BD, sin necesidad de recurrir a sentencias SQL, ya que son funciones que interactúan directamente con MySql. Sin embargo Php también permite ejecutar sentencias SQL que el programador desee realizar, a fin de consultar, actualizar, insertar y borrar los datos que se almacenan en el dicho servidor.

A continuación describiremos algunas de estas funciones. Cabe resaltar que estas funciones se usan cuando ya se estableció una conexión con MySql y tenemos el identificador de enlace de la misma.

- *Crear y Borrar Bases de Datos*

```
mysql_create_db (nom,$link)
```

donde nom es el nombre de la nueva base de datos y $link el enlace de la conexión. Esta función devuelve TRUE si la base de datos es creada y FALSE si no es posible hacerlo. Si intentamos crear una base de datos con un nombre ya existente la función nos devolverá FALSE.

```
mysql_drop_db (nom,$link)
```

Intenta suprimir una base de datos completa del servidor asociado al identificador de enlace. Devuelve: verdadero si éxito, falso si error.

- *Consulta de nombres de Bases de Datos, Tablas y Campos*

```
int mysql_list_dbs ( $link)
```

Devuelve un puntero de resultado que contiene las bases disponibles en el actual demonio mysql. Se usa el puntero ($res) con la función `mysql_db_name($res,numero)` para recoger el nombre de las BD disponibles.

```
    $p=mysql_list_dbs($link);
    mysql_db_name($p,1)
```

```
int mysql_list_tables ("nombre BD" ,$link)
```

Toma el nombre de la base y devuelve un puntero ($res) de resultado par posterior uso de la función `mysql_tablename($res,numero)` para extraer los nombres de las tablas del puntero.

```
    $p=mysql_list_tables("Facturacion",$link);
    mysql_table_name($p,1)
```

```
mysql_list_fields ("nombre bd", "nombre tabla", $link)
```

Lista información sobre la tabla. Los argumentos son la base de datos y el nombre de la tabla. Devuelve un puntero ($res) que puede ser usado para obtener información por diversas funciones:

```
$p=mysql_list_fields("Facturacion","Producto",$link);
mysql_field_name($p,1) #Nombre del Campo Especificado
mysql_field_len($p,1)  #Longitud del Campo Especificado
mysql_field_flags($p,1)#Flag del Campo Especificado "Null","Primary_key".
mysql_field_type($p,1) #Tipo del Campo Especificado "int","string"..
```

- *Envió de sentencias SQL a MySQL*

```
$res=mysql_query($sent, $link)
```

donde $sent es la sentencia SQL (una cadena) y $link el identificador de conexión. El identificador de resultado que recoge la variable $res es el que será usado por las funciones PHP que nos facilitan información sobre la estructura y los datos del resultado.

```
mysql_num_fields ($res)
```

Está función - en la que $res es el identificador de resultado - devuelve el número de campos del resultado.

```
mysql_num_rows ($res)
```

Devuelve el número de registros que contiene el resultado. Si el mismo no contiene datos devolverá CERO. También se lo puede usar con la variable devuelta ($res) por las sentencias mysql_list_dbs(), mysql_list_tables(), mysql_list_fields () para

obtener el número de tablas, campos y bases de datos correspondientes a cada función.

- *Manejo de Resultado de Sentencias SQL*

Una de las funcionalidades de Php es de poder manejar de una manera sencilla los resultados de las consultas a la Base de Datos. Luego del envío de las sentencias SQL a la Base de Datos esta produce el resultado deseado (en el identificador *$res*) y php usa una función llamada mysql_fetch_row() para manejar los mismos y procesarlos como sea necesarios. Su sintaxis es:

```
mysql_fetch_row ($res)
```

Selecciona una fila de datos del resultado asociado al identificador de resultado $res producido por la sentencia mysql_query (). Cada fila (pueden ser varias) es devuelta como una matriz escalar cuyo primer índice es cero. Devolverá FALSE cuando ya no queden más filas.

```
mysql_fetch_array ($res)
```

Es una versión extendida de mysql_fetch_row. Además de guardar los datos en el índice numérico del array, los guarda también con índices asociativos, usando el nombre del campo como índice.

```
mysql_free_result($res)
```

Esta función sirve para liberar memoria del resultado producido por la sentencia. Esto es útil cuando preocupa el uso de memoria en un script.

```
<?
mysql_connect($host,$user,$password);
$result = mysql_query("select * from table");
while($row = mysql_fetch_array($result)) {
    echo $row["user_id"];
    echo $row["fullname"];
}
mysql_free_result($result);
?>
```

- *Visualizando errores de la Base de Datos*

Php ofrece dos funciones para visualizar errores que ocurren en la interacción con la Base de Datos. Estas funciones lo que hacen es devolver una cadena que se produce por el servidor de base de datos.

```
int mysql_errno($res)
```

Esta función nos devuelve el número de error producido por cierta operación. Este número es propio de la base de datos.

```
string mysql_errno($res)
```

Esta función nos devuelve el texto que del mensaje de error de la última operación MySql.

```
<?php
mysql_connect("localhost","root","root");
echo mysql_errno().": ".mysql_error()."<BR>";
mysql_select_db("nonexistentdb");
echo mysql_errno().": ".mysql_error()."<BR>";
$conn = mysql_query("SELECT * FROM nonexistenttable");
echo mysql_errno().": ".mysql_error()."<BR>";
?>
```

Ejemplo .

Vamos a describir un ejemplo donde interactúa Php usando las funciones antes descritas, con MySql. Este ejemplo administrará una tabla llamada producto, la cual tiene 5 campos: “codigo”, ”nombre”, ”marca”, ”descripción”, ”precio”. Esta tabla deberá estar creada en una base de datos llamada “Tienda “ dentro de MySql con el código:

```
CREATE TABLE `producto` (
  `codigo`        int(3)            unsigned NOT NULL auto_increment,
  `nombre`        varchar(50)       NOT NULL default '',
  `marca`         varchar(50)       default NULL,
  `descripcion`   varchar(250)      default NULL,
  `precio`        flota             default '0',
  PRIMARY KEY  (`codigo`)
)COMMENT='Productos de la Tienda';
```

Se creará una carpeta llamada “tienda” dentro del directorio raíz del servidor web Apache. En esta carpeta crearemos 5 archivos php para mantener la tabla.

- *productos.php*

Este archivo desplegará todos los productos (registros) que están almacenados en las tabla, con la posibilidad de editarlos mediante un link a otra página. Este link se creará sobre cada producto de la tabla con el código de cada uno para saber de cual se trata.

```
<html>
<head>
<title>Mantenimiento de Productos</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<body bgcolor="#FFFFE6" text="#000000">
<div align="center">
<p><b><font face="Verdana, Arial, Helvetica, sans-serif" color="#CC3366"
size="4">SELECCIONE PRODUCTO</font></b></p>
</div>

<table width="40%" border="1" cellpadding="0" cellspacing=0 align="center">
  <?
        # establecemos la conexión con el servidor
        $conexion=mysql_connect("localhost","root","root");

        #asiganamos la conexión a una base de datos determinada
        mysql_select_db("compras",$conexion);

        #Actualizamos el registro
        $resultado = mysql_query("SELECT codigo,nombre FROM producto",$conexion);

        while ($row = mysql_fetch_array($resultado)) {
```

```
            echo "<tr>";
            echo "<td><a
href='editarProducto.php?cod=".$row['codigo']."'>".$row['nombre']."</a></td>";
            echo "</tr>";
        }
?>
</table>
<p> </p>
<font face="Verdana, Arial, Helvetica, sans-serif"><font size="2"><a
href="ingresoProducto.php">ingresar nuevo Producto</a></font><BR>
</font>
</body>
</html>
```

- *ingresoProducto.php*

Este archivo despliega un formulario vació con el cual se llamará a otro (ingresoProductoProceso.php) para que ingrese un nuevo producto a la tabla producto. Utiliza codigo JavaScript para validar el ingreso del nombre del mismo.

```
<html>
<head>
<title>Ingresar Producto</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<SCRIPT language=javascript1.2>
function validar(){
     with(document.formulario)
         if (nombre.value == "" ){
            alert("El nombre del Producto debe ser Ingresado");
            return false;
         }
     return true;
}
</SCRIPT>
<body bgcolor="#FFFFE6" text="#000000">
<p align="center"><b><font face="Verdana, Arial, Helvetica, sans-serif" size="4"
color="#CC3366">INGRESAR NUEVO PRODUCTO</font></b></p>

<form name="formulario" method="post" onSubmit='return validar()'
action="ingresoProductoProceso.php">
  <font face="Verdana, Arial, Helvetica, sans-serif" color="#333399" size='2'>
  <table width="55%" border="1" align="center" bgcolor="#FFFFCC">
    <tr>
      <td width="30%">Nombre</td>
      <td width="70%">
        <input type="text" name="nombre" size="50" maxlength="50">
      </td>
    </tr>
    <tr>
      <td width="30%">Marca</td>
      <td width="70%">
        <input type="text" name="marca" size="50" maxlength="50">
      </td>
    </tr>
    <tr>
      <td width="30%">Descripci&oacute;n</td>
      <td width="70%">
        <input type="text" name="descripcion" size="50" maxlength="250">
      </td>
    </tr>
    <tr>
      <td width="30%">Precio</td>
      <td width="70%">
        <input type="text" name="precio" size="12" maxlength="15">
      </td>
    </tr>
```

```
    <tr>
      <td colspan="2" align="center">
        <input type="submit" name="grabar" value="Grabar">
        <input type="reset" name="reset" value="Limpiar">
      </td>
    </tr>
  </table>
  </font>
</form>
</body>
</html>
```

- *ingresoProductoProceso.php*

Esta página es llamada luego de ingresar los datos de un nuevo registro, en esta se conecta a la BD para ingresar nuestro producto con sentencia SQL y luego se redirecciona a la página principal de productos.

```
<?
//Verifica si esta desea ingresar un nuevo Producto
if(isset($HTTP_POST_VARS['grabar'])){
   //recojo las Variables
   $v1 =  $HTTP_POST_VARS['nombre'];
   //si no ingresaron variables las inicializo a vacío
   (isset($HTTP_POST_VARS['marca']))?$v2=$HTTP_POST_VARS['marca']:$v2="";

(isset($HTTP_POST_VARS['descripcion']))?$v3=$HTTP_POST_VARS['descripcion']:$v3="";
   (isset($HTTP_POST_VARS['precio']) && $HTTP_POST_VARS['precio'] !=
"")?$v4=$HTTP_POST_VARS['precio']:$v4="0";

   $insert="INSERT INTO producto (nombre,marca";
   $insert .=",descripcion,precio) ";
   $insert .="VALUES('$v1','$v2','$v3',$v4)";

   # establecemos la conexion con el servidor
   $conexion=mysql_connect("localhost","root","root");

   #asiganamos la conexión a una base de datos determinada
   mysql_select_db("compras",$conexion);

   # Insertamos el registro
   $resultado= mysql_query($insert,$conexion);
}
header("Location: productos.php");
exit();
?>
```

- *editarProducto.php*

Este archivo recibe el código de producto a editar vía URL, es decir con el método get. Luego este se conecta para desplegarlos datos del registro y posteriormente llama a una página para actualizarlos o eliminar el producto dependiendo del botón presionado.

```
<html>
<head>
<title>Editar Datos del Producto</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<SCRIPT language=javascript1.2>
function borrarProducto(){
       if(confirm("Está seguro en eliminar este Producto"))
       {
              return true;
```

```
       }
       return false;
}
function validar(){
     with(document.formulario)
         if (nombre.value == "" ){
            alert("El nombre del Producto debe ser Ingresado");
            return false;
         }
     return true;
}
</SCRIPT>
<?
if(isset($HTTP_GET_VARS['cod'])){ //Actualiza algun Producto
//recoje las Variables
$cod = $HTTP_GET_VARS['cod'];

# establecemos la conexion con el servidor
$conexion=mysql_connect("localhost","root","root");
#asiganamos la conexión a una base de datos determinada
mysql_select_db("compras",$conexion);
# Selecionamos el registro
$resultado= mysql_query("SELECT * FROM producto WHERE codigo=$cod",$conexion);
#asigno el registro a las variables
while ($row = mysql_fetch_array($resultado)) {
       $nombre = $row['nombre'];
       $marca = $row['marca'];
       $descripcion = $row['descripcion'];
       $precio = $row['precio'];
}
?>
<body bgcolor="#FFFFE6" text="#000000">
<p align="center"><b><font face="Verdana, Arial, Helvetica, sans-serif" size="4"
color="#CC3366">ACTUALIZACI&Oacute;N DATOS PRODUCTO</font></b></p>

<form name="formulario" method="post" onSubmit='return validar()'
action="editarProductoProceso.php?cod=<?echo $cod;?>">
  <font face="Verdana, Arial, Helvetica, sans-serif" color="#333399" size='2'>
  <table width="55%" border="1" align="center" bgcolor="#FFFFCC">
    <tr>
      <td width="30%">Nombre</td>
      <td width="70%">
        <input type="text" name="nombre" value="<?= $nombre;?>" size="50"
maxlength="50">
      </td>
    </tr>
    <tr>
      <td width="30%">Marca</td>
      <td width="70%">
        <input type="text" name="marca" value="<?= $marca;?>" size="50"
maxlength="50">
      </td>
    </tr>
    <tr>
      <td width="30%">Descripci&oacute;n</td>
      <td width="70%">
        <input type="text" name="descripcion" value="<?= $descripcion;?>"
size="50" maxlength="250">
      </td>
    </tr>
    <tr>
      <td width="30%">Precio</td>
      <td width="70%">
        <input type="text" name="precio" value="<?= $precio;?>" size="12"
maxlength="15">
      </td>
    </tr>
    <tr>
      <td colspan="2" align="center">
        <input type="submit" name="grabar" value="Grabar">
```

```
        <input type="submit" name="eliminar" onClick='return(borrarProducto())'
value="Eliminar">
      </td>
    </tr>
  </table>
  </font>
</form>
<?
}else{
      echo "<H1>No existe ese código de Producto</H1>!";
}
?>
</body>
</html>
```

- *editarProductoProceso.php*

Esta página procesa la petición ya sea para actualizar el producto con nuevos datos o eliminarlo de la tabla.

```
<?
if(isset($HTTP_GET_VARS['cod'])){ //Inserta un nuevo Doctor
    //recojo las Variables
    $cod = $HTTP_GET_VARS['cod'];

    //Reviso si desea actualizar o borrar un Producto
    if(isset($HTTP_POST_VARS['grabar'])){ //Actualiza Producto
       //recojo las Variables
       $v1 =  $HTTP_POST_VARS['nombre'];
       //si no ingresaron variables las inicializo a vacío
       (isset($HTTP_POST_VARS['marca']))?$v2=$HTTP_POST_VARS['marca']:$v2="";
(isset($HTTP_POST_VARS['descripcion']))?$v3=$HTTP_POST_VARS['descripcion']:$v3="";
(isset($HTTP_POST_VARS['precio']) && $HTTP_POST_VARS['precio']
!="")?$v4=$HTTP_POST_VARS['precio']:$v4="0";

       $update="UPDATE producto SET nombre='$v1',marca='$v2'";
       $update .=",descripcion='$v3',precio=$v4 ";
       $update .="WHERE codigo=$cod";

       # establecemos la conexion con el servidor
       $conexion=mysql_connect("localhost","root","root");
       #asiganamos la conexión a una base de datos determinada
       mysql_select_db("compras",$conexion);
       #Actualizamos el registro
        if(!mysql_query($update,$conexion))
        {
              echo "algo paso";
              echo mysql_errno().": ".mysql_error()."<BR>";
              echo "<BR>SQL=$update";
              exit();
       }

   }elseif(isset($HTTP_POST_VARS['eliminar'])){  //Elimina Producto

       $delete="DELETE FROM producto WHERE codigo=$cod";
       # establecemos la conexión con el servidor
       $conexion=mysql_connect("localhost","root","root");

       #asiganamos la conexión a una base de datos determinada
       mysql_select_db("compras",$conexion);

       #Actualizamos el registro
       mysql_query($delete,$conexion);
   }
}
header("Location: productos.php");
exit();
?>
```

ANEXOS

Configuración del servidor Apache para soportar PHP y MySql

Los programas se deben bajar de los sitios en Internet:

www.apache.org
www.php.net
www.mysql.com

Se instalan cada software siguiendo las instrucciones de los proveedores y se configuran los siguientes archivos:

Apache

Archivo httpd.conf (ubicación c:*ruta de instalación*\apache\conf\ )

Se deben modificar las líneas:

```
ServerName localhost
DocumentRoot "C:/ruta de instalación/Apache/htdocs"
Port 8080   #el número dependerá de la máquina

ScriptAlias /php/ "C:/PHP/"   #directorio de instalación de PHP
ScriptAlias /php4/ "C:/PHP/"  #directorio de instalación de PHP

AddType application/x-httpd-php4 .php
AddType application/x-httpd-php .php .php4
AddType application/x-httpd-php-source .phps

AddHandler php-script .php .php4

Action php-script /php/php.exe
Action application/x-httpd-php4 "/php/php.exe"
```

Php

Archivo php.ini (ubicación c:\Windows)

```
doc_root = "C:\ruta de instalación\Apache\htdocs"
```

MySql

Sólo es necesario configurar la clave del root al iniciar el programa por primera vez bajo el sistema operativo windows con el archivo:

```
C:\mysql\bin\winmysqladmin.exe
```

Luego aparecerá un icono de un semáforo en verde en la barra de tareas de Windows indicando si el servidor MySql esta levantado.