

Manual de PHP y MySQL

Manual de PHP y MySQL

Luis Felipe Wanumen Silva
Darín Jairo Mosquera Palacios
Laura Ximena García Vaca



Agradecimientos especiales a Dios Todopoderoso, a nuestras madres por su paciencia cada vez que nos veían trabajando en la elaboración de este manual y a todas aquellas personas a quienes no les pudimos dedicar tiempo por estar desarrollando este material.

Agradecimientos a los profesores investigadores de la Universidad Distrital, que nos han apoyado en la realización de este trabajo. De manera especial queremos dar las gracias a nuestros coordinadores y amigos por su buena energía. Es importante agradecer a nuestros coordinadores de proyectos curriculares, quienes nos han dado la posibilidad de incluir en nuestros planes de trabajo horas para la escritura de documentos de investigación. Estas horas, a pesar de haber sido bien aprovechadas, no fueron suficientes y por esta razón se reitera nuestros agradecimientos a nuestras familias, quienes han comprendido esta situación y han aceptado que les quitémos tiempo que muy seguramente recuperaremos.



UD
Editorial

E2
ESPACIOS

© Universidad Distrital Francisco José de Caldas
© Luis Felipe Wanumen Silva, Darín Jairo Mosquera Palacios,
Laura Ximena García Vaca
Primera edición, agosto de 2017
ISBN: 978-958-5434-58-5

Dirección Sección de Publicaciones
Rubén Eliécer Carvajalino C.

Coordinación editorial
Nathalie De La Cuadra N.

Corrección de estilo
Josefina Marambio Márquez

Diagramación
Cristina Castañeda Pedraza

Editorial UD
Universidad Distrital Francisco José de Caldas
Carrera 24 No. 34-37
Teléfono: 3239300 ext. 6202
Correo electrónico: publicaciones@udistrital.edu.co

Manual de PHP y MySQL / Luis Felipe Wanumen Silva y
otros. -- Bogotá: Universidad Distrital Francisco José
de Caldas, 2017.
194 páginas ; 24 cm.
ISBN 978-958-5434-58-5
1. Ingeniería de sistemas 2. Lenguajes de programación
(Computadores) 3. MySQL (Lenguaje de programación de
computadores) 4. PHP (Lenguaje de programación de
computadores) I. Wanumen Silva, Luis Felipe, autor.
005.133 cd 21 ed.
A1576885
CEP-Banco de la República-Biblioteca Luis Ángel Arango

Todos los derechos reservados.

Esta obra no puede ser reproducida sin el permiso previo escrito de la
Sección de Publicaciones de la Universidad Distrital.
Hecho en Colombia

Contenido

Introducción	11
Prefacio	13
Cosas básicas del lenguaje PHP	15
Distinguir código PHP de código HTML	15
Obtener caracteres de una cadena	20
Establecer la longitud de una cadena	21
<i>Arrays en PHP</i>	23
Arrays escalares	23
Recorrer arrays asociativos	25
Matrices en PHP	27
Definición	27
Ejercicio simple de matrices	27
Ejercicio para aclarar el concepto de matrices	29
El tipo no importa en las matrices	30
Características de orientación a objetos de PHP	43
Sobre el lenguaje PHP	43
Comparar PHP con ASP 6.0	43
Bases de datos soportadas con PHP	44
Utilidades para instalar ambientes integrados con PHP	45
Error común al inicializar variables en clases con PHP	47
Desarrollar aplicaciones PHP con bases de datos	67
El servidor de bases de datos MySQL	67
La conexión con el servidor de bases de datos MySQL	71
La conexión con la base de datos ubicada en el servidor de bases de datos	72
Eliminación de registros con PHP y MySQL	94
Parámetros y formularios con PHP	99
Pasar parámetros en formularios con el método POST	99
Pasar parámetros en formularios con el método GET	102
Pasar parámetro con botones e hiperenlaces	105
Pasar varios parámetro con botones	110

Ejemplo de una búsqueda	131
Objetivo general	131
Objetivo específico	131
Metodología	131
Inserción de datos validados	141
Objetivo general	141
Objetivo específico	141
Metodología	141
Manejo de archivos	145
Objetivo general	145
Objetivo específico	145
Metodología	145
Cookies con PHP	157
Objetivo general	157
Objetivos específicos	157
Metodología	157
Conceptos básicos de cookies	158
Crear y recuperar cookies sencillas	158
Usar la cookie para un login	162
Programas de elementos de interfaz gráfica	165
Un combo que depende de otro combo	165
Paginación en PHP	171
Objetivo general	171
Objetivos específicos	171
Metodología	171
Recursos	171
Bibliografía	172
Desarrollo del contenido de la clase	172
Interoperar entre MySQL y otros motores	185
Instalar ODBC para MySQL	185
Importar datos de MySQL hasta Access	188
Referencias	191

Introducción

El manual de PHP ofrece al lector una exposición clara de los conceptos introductorios a la programación en PHP. Para el desarrollo de este se supone que el lector ya se encuentra familiarizado con los conceptos básicos de programación. Se dedican especiales esfuerzos a explicar estos conceptos básicos de programación en PHP, a la par que se realizan ejercicios con *arrays*, matrices y accesos a bases de datos en MySQL.

El libro muestra en forma pedagógica aspectos básicos, y a medida que se avanza en el texto, la complejidad de los ejercicios se incrementa. Es importante mencionar que los conceptos mostrados en el presente manual son aplicables a cualquier otro lenguaje de programación web. En este sentido, el uso práctico de este libro es importante. Por otra parte, es importante mencionar que al finalizar el estudio de este material, el estudiante podrá crear aplicaciones sencillas con PHP.

No se puede afirmar que aplicaciones complejas se pueden hacer con la lectura de este libro, ya que el nivel de este documento es básico y está dirigido a estudiantes que se están iniciando en este lenguaje. Si ya se tiene algún nivel de conocimiento en otro lenguaje de programación, posiblemente se encontrará que muchas de las cosas de este material son similares a otros lenguajes de programación. Sin embargo, la sintaxis y las especificidades del lenguaje PHP lo hacen un documento de consulta útil para todos los estudiantes de carreras relacionadas con el desarrollo de aplicaciones o para cualquier otro profesional de las tecnologías de la información.

Prefacio

El capítulo “Cosas básicas del lenguaje PHP” se destina a presentar lo básico del lenguaje PHP, para que al finalizar el capítulo el estudiante pueda manejar la impresión de textos y el manejo básico de variables.

El segundo capítulo, “*Arrays* en PHP”, explica en detalle un aspecto fundamental de los *arrays* en PHP y presenta *arrays* asociativos y *arrays* escalares.

El capítulo “Matrices en PHP” muestra al detalle aspectos de manejo de matrices.

En “Características de orientación a objetos de PHP” se desarrolla el manejo de los principios de orientación a objetos en PHP, donde se muestra el manejo del polimorfismo, de la herencia y de una serie de aspectos propios de los lenguajes que soportan objetos.

En el capítulo “Desarrollando aplicaciones PHP con bases de datos” se entregan las bases para que el estudiante elabore programas en PHP que accedan a bases de datos hechas en MySQL. La lógica de programación puede aplicarse a cualquier otro motor de bases de datos, siempre y cuando se cuenten con las clases y los drivers que permiten la conexión con otros motores de bases de datos.

En “Parámetros y formularios con PHP” se profundiza en el manejo de formularios, un aspecto fundamental a la hora de hacer aplicaciones, ya que con ellos se consigue capturar información que posteriormente va a ser enviada a servidores y en muchas ocasiones comparada o almacenada con la información existente en las bases de datos.

En el capítulo “Ejemplo de una búsqueda” se puede apreciar el desarrollo de un primer ejercicio en PHP, que permite buscar un registro almacenado previamente en la base de datos MySQL. El ejemplo es bastante didáctico, pues se parte de cómo se usan los comandos en la base de datos y posteriormente cómo se hace la aplicación que interactúa con dicha base de datos.

En “Inserción de datos validados” se explica cómo se hacen inserciones en una base de datos, pero validando que estos sean coherentes antes de enviarlos a la base de datos.

En el capítulo “Manejo de archivos” se realiza un ejercicio que enseña a leer, crear y escribir en archivos planos desde aplicaciones PHP, así como incluirlos en las aplicaciones PHP.

En “*Cookies* con PHP” se enseña el concepto de *cookies*, uno de los más importantes a la hora de hacer aplicaciones seguras en cualquier lenguaje de programación web.

En el capítulo “Programas de elementos de interfaz gráfica” se muestran ejemplos de programas que aprovechan las funcionalidades del lenguaje PHP mezclado con el lenguaje HTML para mostrar elementos de interfaz gráfica HTML que interactúan entre sí.

En “Paginación en PHP” se intenta mostrar cómo la paginación de resultados disminuye la cantidad de registros que se le entregan visualmente a un usuario final, de tal suerte que este no se siente inundado con tanta información. Obviamente este tipo de técnicas son útiles cuando el número de registros en una aplicación es grande. Esta técnica no solo disminuye el número de registros en una página, sino que mejora la experiencia de usuario al mismo tiempo que disminuye el tráfico de red.

En el último capítulo, “Interoperar entre MySQL y otros motores”, se explica que la interoperabilidad entre MySQL y otros motores de bases de datos es algo que inquieta a muchas personas, sobre todo a aquellas que mantienen aplicaciones en otros motores y están apenas experimentando con MySQL. La razón para que esto suceda no es única y puede deberse a factores tales como la posibilidad de ver qué tan confiable es volver a tomar las aplicaciones que antes se tenían en otros motores, o de pronto ver la compatibilidad de MySQL con otros manejadores. De todas maneras, sea cual sea la razón por la cual el tema interese, es fundamental que el estudiante tenga una idea de la importancia de interoperar entre MySQL y otros motores de bases de datos.

En este capítulo se muestran al lector los aspectos básicos del lenguaje PHP, por lo que es una sección obligada para los principiantes. Por supuesto, si usted es una persona

Cosas básicas del lenguaje PHP ---

que ha trabajado bastante con PHP no necesita leer este capítulo y puede saltar a las siguientes secciones, que tienen temas que serán de gran interés.

Distinguir código PHP de código HTML

El código PHP debe ir necesariamente en páginas con la extensión “php”. En el pasado se utilizó la extensión “phtml”, pero la verdad es que este tipo de extensiones para PHP están bastante abolidas.

Es posible que se creen páginas PHP que en realidad no tengan código PHP, sino código HTML o JavaScript, lo que es perfectamente válido. Lo que sucede es que despistaría al navegante, puesto que este pensaría que la página está procesando sentencias y le está mostrando el resultado de la ejecución de código PHP. Como conclusión de esto, podemos decir que las páginas HTML bien formadas y que cumplan con los estándares correrán si se les cambia la extensión “htm” o “html” por “php”.

De todas maneras, lo que no es posible hacer es colocar en una página que tenga extensión “htm” o “html” código PHP, debido a que esto acarrearía que salgan algunos errores y que se muestre dicho código en el lado del cliente.

El otro caso que más comúnmente se presenta es el de páginas PHP que tienen código PHP y código HTML. Obviamente podemos tener páginas PHP que única y exclusivamente tengan código PHP. Para no desviarnos del tema que estamos tratando, pensemos que se tiene una página con código PHP y con código HTML. Surge entonces la pregunta: ¿cómo distinguir el código PHP del código HTML, o de otro código que se encuentre en dicha página? La respuesta es bien sencilla: mediante unos identificadores de inicio y finalización de código PHP.

Estos identificadores son:

Identificador de inicio de código PHP	Identificador de finalización de código PHP
<?	?>
<?php	?>
<script language="php">	</script>
<%	%>

Es importante anotar que si se usa un identificador de inicio de código PHP de los mostrados en la tabla anterior, se debe utilizar el identificador de finalización mostrado al frente del mismo en la tabla.

Pero como imaginamos que el lector entiende mejor los conceptos con ejercicios prácticos, veamos el código de la siguiente página:

```
<HTML>
<HEAD>
</HEAD>
<BODY>
<?php
// Todo lo que esté en medio de estos identificadores es código PHP
?>
</BODY>
<HTML>
```

Cabe anotar que el código PHP se puede colocar en cualquier parte de un documento, es decir, la anterior página se pudo haber colocado de la siguiente forma:

```
<HTML>
<HEAD>
<?php
// Todo lo que esté en medio de estos identificadores es código PHP
?>
</HEAD>
<BODY> </BODY>
<HTML>
```

Se pudo haber colocado antes de cualquier código HTML, como se muestra a continuación:

```
<?php
// Todo lo que esté en medio de estos identificadores es código PHP
?>
<HTML>
<HEAD>
</HEAD>
<BODY>
</BODY>
<HTML>
```

O se pudo haber colocado al final del documento:

```
<HTML>
<HEAD>
</HEAD>
<BODY>
</BODY>
<HTML>
<?php
// Todo lo que esté en medio de estos identificadores es código PHP
?>
```

No está de más recordar al lector que a cualquiera de las páginas mostradas se les debió dar un nombre que tenga extensión “php”, porque tiene código PHP en alguna parte de su interior.

Un consejo para el lector es que utilice los identificadores de inicio y terminación de PHP que utilizamos antes, debido a que si escoge otros es posible que le toque configurar algunas cosas, y si es un principiante es mejor evitar la fatiga para que no se desmoralice.

Los comentarios en PHP

Básicamente existen dos tipos de comentarios en PHP: los que cubren una línea exclusivamente y los que sirven para comentar varias líneas de código. A continuación se muestran los comentarios de una línea:

```
<?php
// Soy un comentario de una línea
?>
```

Y los comentarios de varias líneas:

```
<?php
/* Soy un comentario que abarca varias
líneas y finalizo solamente hasta que
encuentre el signo asterisco y la rayita
acostada hacia adelante
*/
?>
```

Las cadenas en PHP

Al igual que con otros tipos de variables en PHP, no se declara el tipo de datos de la variable, y los nombres de estas siempre deben estar precedidos del signo “\$”. Veamos, por ejemplo, una página PHP que genera un error:

```
<?php
int $numero;
?>
```

La razón es sencilla: en PHP no se debe declarar el tipo de datos de la variable, sino que este se deduce cuando a dicha variable se le asignan valores. A manera ilustrativa, la página anterior se puede arreglar de la siguiente manera:

```
<?php
$numero;
?>
```

El lector estará pensando: ¿cómo es posible que el autor hable de tipos de datos enteros cuando el título de la sección habla de cadenas? La respuesta es sencilla. Lo que pasa es que es necesario comprender el manejo de las variables en PHP para explicar el manejo de las variables tipo cadena, que son las que podrían presentarle dificultad de manejo a las personas novatas en PHP.

En la siguiente página se declara una variable y se le asigna una cadena, con lo cual es como si la variable hubiera sido declarada como cadena. Veamos:

```
<?php
$cadena;
$cadena = “Luis Felipe Wanumen Silva”;
?>
```

Como el lector puede observar, la página anterior no muestra resultados en pantalla, debido a que crea una variable y le asigna una cadena a dicha variable, pero no se imprime el valor.

Pero antes de terminar esta sección, veamos el código de la siguiente página:

```
<?php
$cadena;
$cadena = “ ‘Luis’ ‘Felipe’ ‘Wanumen’ ‘Silva’ “;
?>
```

La anterior página le asigna a la variable “\$cadena” el nombre “Luis Felipe Wanumen Silva”, pero observe que dicho nombre tiene unas comillas sencillas y esto también es válido en PHP. Observemos una variante a la página PHP tal como se muestra a continuación:

```
<?php
$cadena;
$cadena = ‘ “Luis” “Felipe” “Wanumen” “Silva” ‘;
?>
```

En esta ocasión se asigna un nombre que tiene comillas sencillas como caracteres que identifican el comienzo y la finalización de la cadena. En otras palabras, si se inicializa una variable asignándole una cadena, dicha cadena puede comenzar con comillas dobles o con comillas sencillas, pero se debe finalizar con el mismo tipo de comillas con que se inició la cadena.

Concatenar cadenas en PHP

En la sección anterior veíamos como crear cadenas, y en esta vamos a ver cómo se concatenan cadenas.

```
<?php
$cad1 = ‘ Luis ‘;
$cad2 = ‘ Felipe ‘;
$cad3 = ‘ Wanumen ‘;
$cad4 = ‘ Silva ‘;
$cad5 = $cad1 . $cad2 . $cad3 . $cad4;
echo $cad5;
?>
```

Como el lector puede observar, la concatenación de cadenas se puede hacer mediante el punto. El cuidado que se debe tener al concatenar es tener en cuenta que el punto quede separado de las dos cadenas que se están concatenando.

El resultado de la ejecución de la anterior página es el siguiente:



La siguiente página muestra otra forma de concatenar cadenas:

```
<?php
$cad1 = ' Luis ';
$cad2 = ' Felipe ';
$cad1 .= $cad2;
echo $cad1;
?>
```

El resultado de la ejecución de la anterior página es similar al siguiente:



Obtener caracteres de una cadena

Las cadenas en realidad se comportan como *arrays* en PHP, con lo cual después de haberle asignado una cadena a una variable, se puede utilizar esta última para hacer referencia a cada uno de los caracteres de dicha cadena. En realidad, la siguiente página asigna una cadena a la variable “\$cad”, y por medio de dicha variable se imprime una a una cada una de las letras que conforman el nombre “LUIS”. Veamos:

```
<?php
$cad = ' Luis ';
$cad1 = $cad[1];
$cad2 = $cad[2];
$cad3 = $cad[3];
$cad4 = $cad[4];
```

```

echo $cad1;
echo $cad2;
echo $cad3;
echo $cad4;
?>

```

Con lo cual obtenemos un resultado similar al siguiente al ejecutar la anterior página:



Establecer la longitud de una cadena

Para establecer la longitud de una cadena utilizamos la función “strlen”. La siguiente página hace uso de dicha función para mostrar la longitud de la cadena conformada por el nombre “Luis Felipe Wanumen Silva”. Veamos:

```

<?php
$cad = 'Luis Felipe Wanumen Silva';
$longitud = strlen($cad);
echo "La longitud de la cadena es: " . $longitud;
?>

```

Con lo cual el resultado de la ejecución de la anterior página es similar al siguiente:



Arrays en PHP

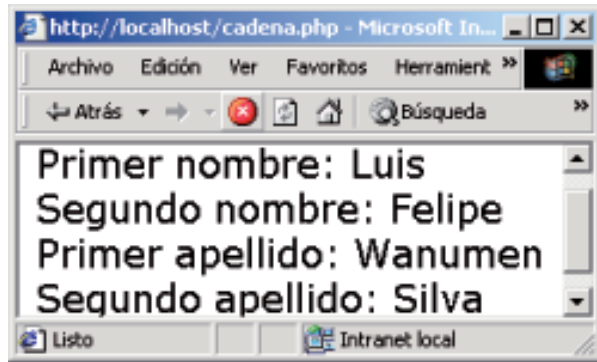
Arrays escalares

Veamos la siguiente página:

```
<HTML>
<BODY>
<?php
$x[1] = 'Luis<BR>';
$x[2] = 'Felipe<BR>';
$x[3] = 'Wanumen<BR>';
$x[4] = 'Silva<BR>';
echo "Primer nombre: " . $x[1];
echo "Segundo nombre: " . $x[2];
echo "Primer apellido: " . $x[3];
echo "Segundo apellido: " . $x[4];
?>
</BODY>
</HTML>
```

Aquí se define un *array* escalar debido a que el índice que se utiliza es un número. En este caso se supone que hay cuatro posiciones de memoria que almacenan datos. Dichas posiciones son accesibles a la hora de imprimirlas combinando el nombre del *array* con el número correspondiente al índice. En este caso se manejan cuatro índices y la verdad es que el tratamiento de este tipo de *arrays* no tiene mayor dificultad, sobre todo para programadores en otros lenguajes de programación. En la próxima sección se mostrará que este no es el único tipo de *arrays* admitidos por PHP.

El resultado de la ejecución de la anterior página es similar al siguiente:

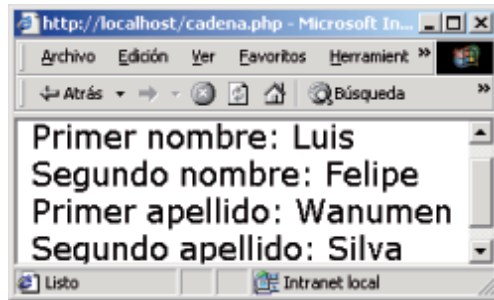


Arrays asociativos

Normalmente los *arrays* tienen unas formas de acceder a las diversas posiciones, lo que se hace por medio de un índice. Lo más interesante de PHP es que además de permitir este tipo de *arrays*, también admite la creación de *arrays* asociativos, tal como se muestra a continuación:

```
<HTML>
<BODY>
<?php
    $x["nombre1"] = 'Luis<BR>';
    $x["nombre2"] = 'Felipe<BR>';
    $x["apellido1"] = 'Wanumen<BR>';
    $x["apellido2"] = 'Silva<BR>';
    echo "Primer nombre: " . $x["nombre1"];
    echo "Segundo nombre: " . $x["nombre2"];
    echo "Primer apellido: " . $x["apellido1"];
    echo "Segundo apellido: " . $x["apellido2"];
?>
</BODY>
</HTML>
```

El resultado de la ejecución de la página anterior es similar al siguiente:



Recorrer *arrays* asociativos

En la sección anterior se explicó cómo se crean *arrays* asociativos y se mencionó que estos no eran lógicamente consecutivos, básicamente debido a su falta de un índice numérico. De todas maneras, a pesar de que no exista un orden lógico, físicamente los elementos quedan almacenados en el orden en el que son asignados.

Es bueno que tengamos presente que para el caso de *arrays* indexados el índice es el número que identifica la posición, pero para el caso de los *arrays* asociativos la cadena que indica la posición es conocida normalmente como “clave” y el contenido del *array* en dicha posición es llamado técnicamente “valor”. Es importante tener presente este concepto para lograr comprender bien las explicaciones que siguen.

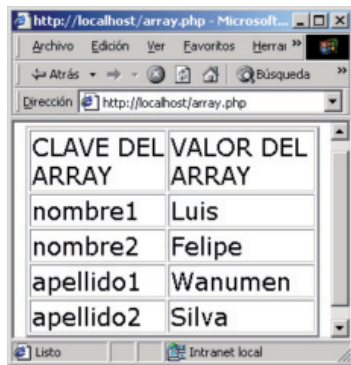
Existe una instrucción en PHP que se podría catalogar como una instrucción de iteración, debido a que permite iterar por cada uno de los elementos de un *array* asociativo. Esta es la instrucción “foreach”, que en español significa “para cada uno”. A continuación se presenta el código fuente de una página PHP que muestra el uso de esta instrucción para recorrer un *array* asociativo. Veamos:

```
<HTML>
<BODY>
<TABLE BORDER=2>
<TR>
<TD>CLAVE DEL ARRAY
</TD>
<TD>VALOR DEL ARRAY
</TD>
</TR>

<?php
$x[“nombre1”] = ‘Luis<BR> ‘;
$x[«nombre2»] = ‘Felipe<BR> ‘;
$x[«apellido1»] = ‘Wanumen<BR> ‘;
$x[«apellido2»] = ‘Silva<BR> ‘;
```

```
foreach($x as $clave => $valor){  
    echo "<TR><TD>";  
    echo $clave;  
    echo "</TD><TD>";  
    echo $valor;  
    echo "</TD></TR>";  
}  
  
?>  
</TABLE>  
</BODY>  
</HTML>
```

La cual produce un resultado similar al siguiente, al ser vista desde el navegador:



CLAVE DEL ARRAY	VALOR DEL ARRAY
nombre1	Luis
nombre2	Felipe
apellido1	Wanumen
apellido2	Silva

Matrices en PHP

No solo las matrices en PHP, sino también en muchos otros lenguajes de programación, son una de las cosas más importantes y básicas que es necesario comprender, debido a que su manejo es vital para el desarrollo de múltiples aplicaciones. En otras palabras, estamos diciendo que conocer la definición, la creación de matrices y su manipulación, es tarea importante que debe proponerse cualquier desarrollador de aplicaciones web con PHP, y en general como desarrollador de *software* en muchos lenguajes de programación.

Definición

Para comprender el concepto de matriz es necesario que el lector comprenda de antemano el concepto de *array*. De esta forma podemos definir una matriz en términos de un *array*.

Ejercicio simple de matrices

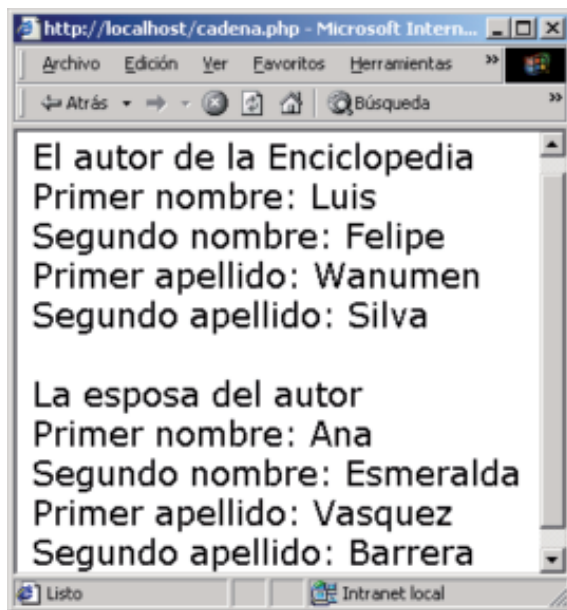
Los *arrays* multidimensionales en realidad son las mismas matrices que se conocen en los lenguajes de programación. La verdad es que PHP es bastante sencillo para manejar este tipo de *arrays*. La siguiente página muestra el uso de una matriz cuya primera fila contiene los nombres al detalle del autor de este manual y la segunda fila contiene los nombres de la esposa del autor. Veamos pues:

```
<HTML>
<BODY>
<?php
$matriz[1][1] = 'Luis<BR>';
$matriz[1][2] = 'Felipe<BR>';
$matriz[1][3] = 'Wanumen<BR>';
$matriz[1][4] = 'Silva<BR>';

$matriz[2][1] = 'Ana<BR>';
```

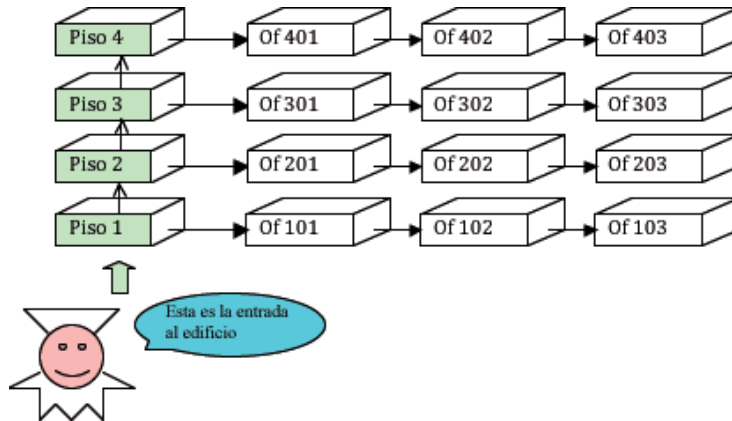
```
$matriz[2][2] = ' Esmeralda<BR> ';  
$matriz[2][3] = ' Vasquez<BR> ';  
$matriz[2][4] = ' Barrera<BR> ';  
  
echo "El autor de la Enciclopedia <BR>";  
echo "Primer nombre: " . $matriz[1][1];  
echo "Segundo nombre: " . $matriz[1][2];  
echo "Primer apellido: " . $matriz[1][3];  
echo "Segundo apellido: " . $matriz[1][4] . "<BR>";  
  
echo "La esposa del autor <BR>";  
echo "Primer nombre: " . $matriz[2][1];  
echo "Segundo nombre: " . $matriz[2][2];  
echo "Primer apellido: " . $matriz[2][3];  
echo "Segundo apellido: " . $matriz[2][4];  
?>  
</BODY>  
</HTML>
```

El resultado de la ejecución de la anterior página PHP es similar al mostrado a continuación:



Ejercicio para aclarar el concepto de matrices

Una matriz es un *array* de *arrays*. En otras palabras, una matriz bidimensional es un conjunto de *arrays* principales, cada uno de ellos permite acceder a otro *array* donde dichos *arrays* se podrían llamar dependientes, y no permiten acceder a más *arrays* para el caso de matrices bidimensionales. Para ilustrar mejor la definición anterior veamos el siguiente esquema:



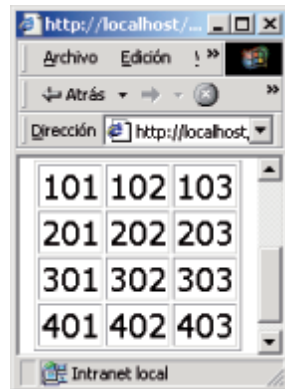
Aquí se muestra cómo entrar a un edificio, esquemáticamente hablando. Se llega a la entrada del edificio, como lo muestra “Luisito”, y de allí se puede acceder a los diversos pisos, tal como lo representan las cajas de color gris. Una vez se posiciona en el piso deseado, se puede acceder a la oficina que se desee llegar.

La siguiente página PHP crea una matriz que sirve para implementar el diagrama anterior. Recordemos que la entrada a los pisos de color gris se maneja con el primer índice “*i*” y el número de oficina se maneja con un segundo índice, que en este caso es la variable “*j*”.

```
<HTML>
<BODY>
<TABLE BORDER = 1>
<?php
$valor = 101;
for($i=1; $i<=4; $i++){
echo "<TR>\n";
for($j=1; $j<4; $j++){
$matriz[$i][$j] = $valor;
echo "<TD>\n";
echo $matriz[$i][$j]."\n";
echo "</TD>\n";
$valor = $valor+1;
}
```

```
}  
echo "</TR>\n";  
$valor = $valor+97;  
echo "<BR>\n";  
}  
?>  
</TABLE>  
</BODY>  
</HTML>
```

Lo anterior produce el siguiente resultado:



El tipo no importa en las matrices

El tipo de datos que se almacenan en una matriz no necesariamente debe ser uno solo, puesto que es posible que se le asignen números a unas posiciones y cadenas a otros, por decir algo. En la siguiente página se muestra y se aclara mediante un ejercicio lo que se está diciendo:

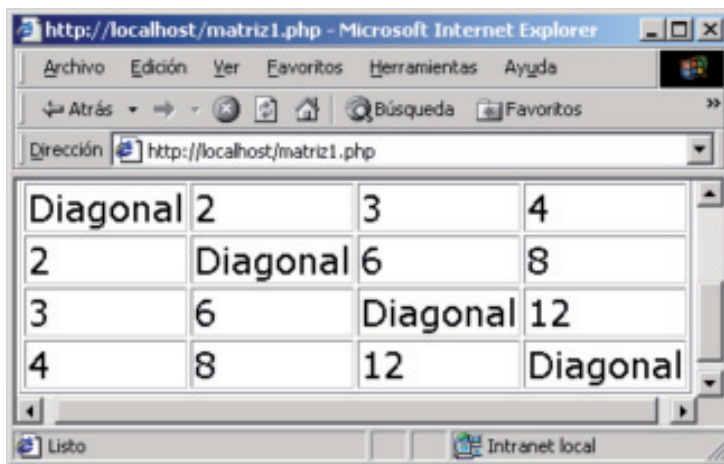
```
<HTML>  
<BODY>  
<TABLE BORDER = 1>  
<?php  
$valor = 101;  
for($i=1; $i<=4; $i++){  
echo "<TR>\n";  
for($j=1; $j<=4; $j++){  
$matriz[$i][$j] = $valor;  
if($i==$j){  
$matriz[$i][$j] = "Diagonal";
```

```

}
else{
$matriz[$i][$j] = $i*$j;
}
echo "<TD>\n";
echo $matriz[$i][$j]."\n";
echo "</TD>\n";
$valor = $valor+1;
}
echo "</TR>\n";
$valor = $valor+97;
echo "<BR>\n";
}
?>
</TABLE>
</BODY>
</HTML>

```

En la anterior página se aprecia que se está creando y llenando una matriz cuadrada de 4 x 4 posiciones. Lo más interesante del ejercicio es notar que los elementos que hacen parte de la diagonal primaria de la matriz, es decir, los elementos cuyo número de fila es igual al número de columna, son llenados con una cadena que concretamente es la cadena "Diagonal". Por otra parte, es bueno observar que a los elementos que no pertenecen a la diagonal primaria se les asigna un número que corresponde exactamente con la multiplicación entre el número de fila y el número de columna a que hace referencia dicha posición. Veamos a manera ilustrativa los resultados generados por la anterior página:



Diagonal	2	3	4
2	Diagonal	6	8
3	6	Diagonal	12
4	8	12	Diagonal

Matrices asociativas

Hasta el momento hemos trabajado con matrices en las cuales se usan los índices para establecer lógicamente qué elemento es el que se desea manipular, es decir, borrar, adicionar o modificar. Por otra parte es bueno que el lector comprenda que de la misma forma como existen *arrays* asociativos —es decir, aquellos que no manejan un índice de tipo numérico, sino una cadena que de ahora en adelante se llamará clave—, también existen matrices que implementan dicha funcionalidad. En realidad cualquier tipo de matriz en PHP permite toda esta flexibilidad.

Lo que sucede con las matrices asociativas es que no existe un orden relativo entre los elementos que existen en la matriz, aunque físicamente los contenidos se guardan en el mismo orden en que fueron insertados. La siguiente página ilustra el uso de las matrices asociativas:

```
<HTML>
<BODY>
<TABLE BORDER = 1>
<?php
$matriz["piso1"]["oficina1"] = 101;
$matriz["piso1"]["oficina2"] = 102;
$matriz["piso1"]["oficina3"] = 103;

$matriz["piso2"]["oficina1"] = 201;
$matriz["piso2"]["oficina2"] = 202;
$matriz["piso2"]["oficina3"] = 203;

$matriz["piso3"]["oficina1"] = 301;
$matriz["piso3"]["oficina2"] = 302;
$matriz["piso3"]["oficina3"] = 303;

$matriz["piso4"]["oficina1"] = 401;
$matriz["piso4"]["oficina2"] = 402;
$matriz["piso4"]["oficina3"] = 403;

echo "<TR>\n";
echo "<TD>\n";
echo $matriz["piso1"]["oficina1"];
echo "</TD>\n";
echo "<TD>\n";
```

```

echo $matriz["piso1"]["oficina2"];
echo "</TD>\n";
echo "<TD>\n";
echo $matriz["piso1"]["oficina3"];
echo "</TD>\n";
echo "</TR>\n";

echo "<TR>\n";
echo "<TD>\n";
echo $matriz["piso2"]["oficina1"];
echo "</TD>\n";
echo "<TD>\n";
echo $matriz["piso2"]["oficina2"];
echo "</TD>\n";
echo "<TD>\n";
echo $matriz["piso2"]["oficina3"];
echo "</TD>\n";
echo "</TR>\n";

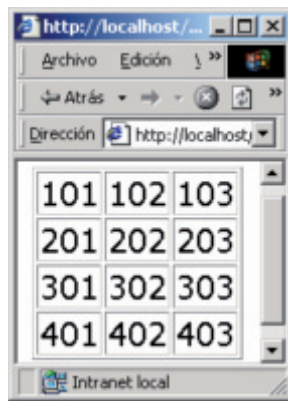
echo "<TR>\n";
echo "<TD>\n";
echo $matriz["piso3"]["oficina1"];
echo "</TD>\n";
echo "<TD>\n";
echo $matriz["piso3"]["oficina2"];
echo "</TD>\n";
echo "<TD>\n";
echo $matriz["piso3"]["oficina3"];
echo "</TD>\n";
echo "</TR>\n";

echo "</TR>\n";
echo "<TD>\n";
echo $matriz["piso4"]["oficina1"];
echo "</TD>\n";
echo "<TD>\n";
echo $matriz["piso4"]["oficina2"];

```

```
echo "</TD>\n";
echo "<TD>\n";
echo $matriz["piso4"]["oficina3"];
echo "</TD>\n";
echo "</TR>\n";
?>
</TABLE>
</BODY>
</HTML>
```

Tal como esperamos, los resultados de la ejecución de la anterior página son similares a los mostrados a continuación:



Valores implícitos en matrices indexadas

Supongamos que tenemos una matriz de 3 x 4, en la cual le asignamos un valor a la posición (3,4). Esto provoca que el resto de posiciones de la matriz tengan asignado el valor “null”, así no se les haya asignado específicamente dicho valor. La siguiente página nos muestra esta situación detalladamente:

```
<HTML>
<BODY>
<TABLE BORDER = 1>
<?php
$valor = 101;
for($i=1; $i<=4; $i++){
echo "<TR>\n";
for($j=1; $j<4; $j++){
if(((($i+$j)%2)==0)){
```



```

$matriz[$i][$j] = $valor;
}
if($matriz[$i][$j]==null){
$matriz[$i][$j] = "Es Nulo";
}
echo "<TD>\n";
echo $matriz[$i][$j]."\n";
echo "</TD>\n";
$valor = $valor+1;
}
echo "</TR>\n";
$valor = $valor+97;
echo "<BR>\n";
}
?>
</TABLE>
</BODY>
</HTML>

```

Para graficar, el lector se puede imaginar un tablero de ajedrez de 3 x 4 (bueno, en realidad no existen tableros de ajedrez de este tamaño, pero el lector comprende):



Si observamos los índices de las posiciones tenemos los siguientes valores:

1 1	1 2	1 3
2 1	2 2	2 3
3 1	3 2	3 3
4 1	4 2	4 3

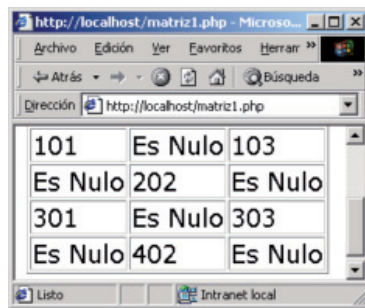
La pregunta que nos formulamos es: ¿qué tienen en común todas las posiciones que corresponden con los cuadros negros de la tabla anterior? La respuesta la encontramos si sumamos los índices de las posiciones tal como se muestra en la siguiente tabla:

1+1=2	1+2=3	1+3=4
2+1=3	2+2=4	2+3=5
3+1=4	3+2=5	3+3=6
4+1=5	4+2=6	4+3=7

2=par	3=IMPAR	4=par
3=IMPAR	4=par	5=IMPAR
4=par	5=IMPAR	6=par
5=IMPAR	6=par	7=IMPAR

La respuesta es que son posiciones impares, con lo cual sabremos que la pregunta que debemos hacer es si la suma de los índices no es divisible exactamente por dos para el caso de los impares o de las celdas oscuras; y para el caso de las celdas blancas diremos que es si la suma de los índices de las celdas es un número divisible exactamente por dos.

Después de haber realizado las anteriores aclaraciones, veamos el resultado de la ejecución de la página:



Este ejercicio se hizo con el ánimo de comprobar y enseñar que las posiciones que se muestran con el texto “Es Nulo”, en realidad son posiciones que tienen un valor “null”, puesto que dicha cadena es asignada mediante las siguientes instrucciones:

```
if($matriz[$i][$j]==null){
    $matriz[$i][$j] = “Es Nulo”;
}
```

Recorrer matrices asociativas

Una sugerencia que se le hace al lector es que se cerciore que comprendió la sección de este libro que mostraba cómo recorrer *arrays* asociativos. El conocimiento de los *arrays* asociativos es vital para comprender el presente ejercicio. En términos más bien sencillos, podríamos decir que una matriz se puede recorrer anidando sentencia de iteración “foreach”.

Recordemos que el ciclo iterativo “foreach” se utiliza con mucha frecuencia para recorrer *arrays* asociativos, de igual manera deducimos que es posible recorrer matrices asociativas haciendo anidaciones de este ciclo.

Si la matriz es bidimensional, es decir de dos dimensiones, bastará con anidar un ciclo iterativo “foreach”; si la matriz es tridimensional, es decir de tres dimensiones, bastará con anidar dos ciclos iterativos “foreach”.

A continuación se muestra una página que define e inicializa una lista de amigos:

```
<HTML>
<BODY>
<TABLE BORDER=2>
<TR>
<TD>CLAVE DEL ARRAY
</TD>
<TD>SUBCLAVE1
</TD>
<TD>SUBCLAVE2
</TD>
<TD>SUBCLAVE3
</TD>
<TD>SUBCLAVE4
</TD>
</TR>

<?php
$x[“amigo1”][“nombre1”] = ‘ Luis<BR> ‘;
$x[“amigo1”][“nombre2”] = ‘ Felipe<BR> ‘;
$x[“amigo1”][“apellido1”] = ‘ Wanumen<BR> ‘;
$x[“amigo1”][“apellido2”] = ‘ Silva<BR> ‘;

$x[“amigo2”][“nombre1”] = ‘ Ana<BR> ‘;
$x[“amigo2”][“nombre2”] = ‘ Esmeralda<BR> ‘;
$x[“amigo2”][“apellido1”] = ‘ Vasquez<BR> ‘;
$x[“amigo2”][“apellido2”] = ‘ Barrera<BR> ‘;

$x[“amigo3”][“nombre1”] = ‘ Luz<BR> ‘;
$x[“amigo3”][“nombre2”] = ‘ Mireya<BR> ‘;
$x[“amigo3”][“apellido1”] = ‘ Cañon<BR> ‘;
$x[“amigo3”][“apellido2”] = ‘ Beltran<BR> ‘;

$x[“amigo4”][“nombre1”] = ‘ Liliana<BR> ‘;
```

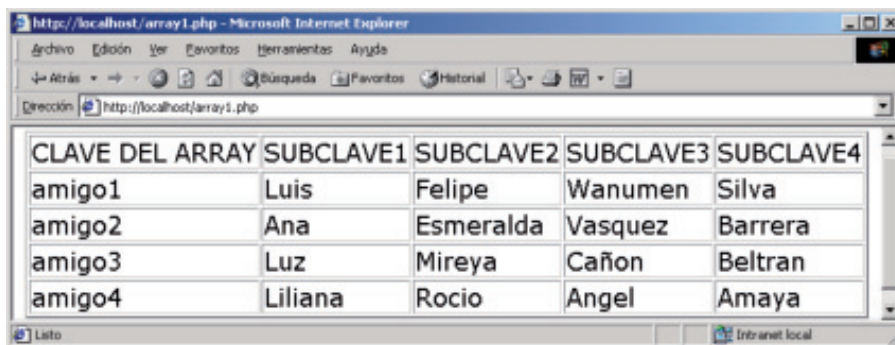
```

$x["amigo4"]["nombre2"] = ' Rocio<BR> ';
$x["amigo4"]["apellido1"] = ' Angel<BR> ';
$x["amigo4"]["apellido2"] = ' Amaya<BR> ';

foreach($x as $clave1=> $vector){
echo "<TR><TD>";
echo $clave1;
echo "</TD>";
foreach($vector as $clave2=> $valor){
echo "<TD>";
echo $valor;
echo "</TD>";
}
echo "</TR>";
}
?>
</TABLE>
</BODY>
</HTML>

```

El resultado de la ejecución de la página anterior es similar al siguiente:



CLAVE DEL ARRAY	SUBCLAVE1	SUBCLAVE2	SUBCLAVE3	SUBCLAVE4
amigo1	Luis	Felipe	Wanumen	Silva
amigo2	Ana	Esmeralda	Vasquez	Barrera
amigo3	Luz	Mireya	Cañon	Beltran
amigo4	Liliana	Rocio	Angel	Amaya

Si no queremos que en la página se muestren esos horribles y molestos títulos en la tabla, sino que se muestre si es el primer nombre, el segundo nombre, el primer apellido o el segundo apellido de nuestros amigos, debemos modificar la página y dejarla como se muestra a continuación:

```

<HTML>
<BODY>
<TABLE BORDER=2>

```

```

<TR>
<TD>CLAVE DEL ARRAY
</TD>
<TD>SUBCLAVE1
</TD>
<TD>SUBCLAVE2
</TD>
<TD>SUBCLAVE3
</TD>
<TD>SUBCLAVE4
</TD>
</TR>

<TR>
<TD>NUMERO DEL AMIGO
</TD>

<?php
$x["amigo1"]["nombre1"] = ' Luis<BR> ';
$x["amigo1"]["nombre2"] = ' Felipe<BR> ';
$x["amigo1"]["apellido1"] = ' Wanumen<BR> ';
$x["amigo1"]["apellido2"] = ' Silva<BR> ';

$x["amigo2"]["nombre1"] = ' Ana<BR> ';
$x["amigo2"]["nombre2"] = ' Esmeralda<BR> ';
$x["amigo2"]["apellido1"] = ' Vasquez<BR> ';
$x["amigo2"]["apellido2"] = ' Barrera<BR> ';

$x["amigo3"]["nombre1"] = ' Luz<BR> ';
$x["amigo3"]["nombre2"] = ' Mireya<BR> ';
$x["amigo3"]["apellido1"] = ' Cañon<BR> ';
$x["amigo3"]["apellido2"] = ' Beltran<BR> ';

$x["amigo4"]["nombre1"] = ' Liliana<BR> ';
$x["amigo4"]["nombre2"] = ' Rocio<BR> ';
$x["amigo4"]["apellido1"] = ' Angel<BR> ';

```

```
$x["amigo4"]["apellido2"] = ' Amaya<BR> ';

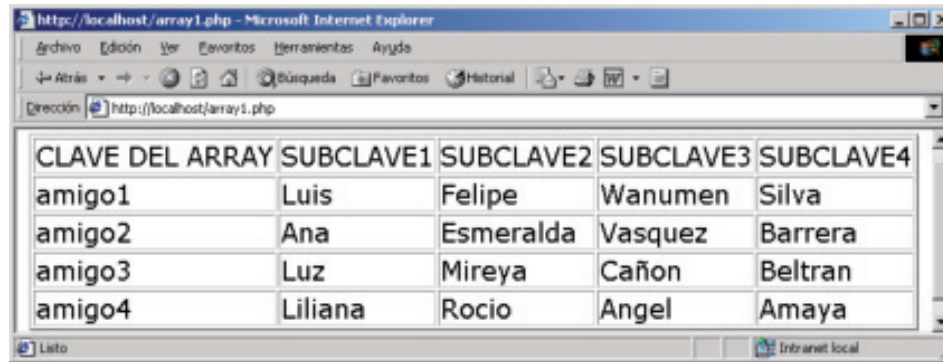
foreach($x as $clave1=> $vector){
    foreach($vector as $clave2=> $valor){
        echo "<TD>";
        echo $clave2;
        echo "</TD>";
        break;
    }
}

?>

</TR>

<?php
foreach($x as $clave1=> $vector){
    echo "<TR><TD>";
    echo $clave1;
    echo "</TD>";
    foreach($vector as $clave2=> $valor){
        echo "<TD>";
        echo $valor;
        echo "</TD>";
    }
    echo "</TR>";
}
?>
</TABLE>
</BODY>
</HTML>
```

Con lo cual se generaría un resultado muy similar al siguiente:



http://localhost/array1.php - Microsoft Internet Explorer

Archivo Edición Ver Favoritos Herramientas Ayuda

← Atrás → Búsqueda Favoritos Historial

Dirección http://localhost/array1.php

CLAVE DEL ARRAY	SUBCLAVE1	SUBCLAVE2	SUBCLAVE3	SUBCLAVE4
amigo1	Luis	Felipe	Wanumen	Silva
amigo2	Ana	Esmeralda	Vasquez	Barrera
amigo3	Luz	Mireya	Cañon	Beltran
amigo4	Liliana	Rocio	Angel	Amaya

Leído Intranet local

Características de orientación a objetos de PHP

Sobre el lenguaje PHP

La programación en el lenguaje PHP se puede hacer usando las características de programación orientada a objetos y las características de la programación estructurada. En esta sección se muestra una aproximación a estos dos tipos de programación.

La programación en PHP en muchas ocasiones da lugar al desorden, debido a que básicamente es posible hacer aplicaciones PHP en las cuales el código que permite mostrar la interfaz de usuario está revuelto, por decirlo de alguna manera, con el código que hace la conexión con una determinada base de datos.

Es importante que conozcamos las primeras instrucciones que acercan a PHP a un lenguaje orientado a objetos. Hemos dicho “acercan” porque no se está afirmando que PHP sea un lenguaje completamente orientado a objetos.

Comparar PHP con ASP 6.0

ASP	PHP
<i>Software</i> de Microsoft.	<i>Software</i> libre.
Solamente corre en plataformas Microsoft, aunque existe un proyecto denominado Mono, mediante el cual prometen también correr páginas ASP, pero todavía se tienen muchos inconvenientes.	Funciona en múltiples plataformas.
En algunas aplicaciones no funciona tan rápido.	Funciona más rápido que ASP.
Lenguaje totalmente estructurado.	Un lenguaje que aunque no es cien por ciento orientado a objetos, incluye algunas características de los lenguajes orientados a objetos, entre ellos la capacidad para permitir la creación de clases y la herencia (aunque no permite la herencia múltiple y otras cuestiones de los lenguajes totalmente orientados a objetos).

El lenguaje es a veces limitado.	Es un lenguaje más potente.
Licencia cerrada.	Licencia abierta.
En la actualidad solamente está probado con el servidor Internet Information Server, y por supuesto, con los servidores de desarrollo en estaciones tales como el personal web server.	<p>PHP funciona perfectamente con nueve servidores http incluyendo el servidor de Internet de Microsoft. Para no ir tan lejos con este comparativo, a continuación se muestra un listado de los servidores con los cuales PHP es compatible:</p> <ul style="list-style-type: none"> •Apache (UNIX, Win32) •Internet Information Server •CGI •fhttpd •ISAPI (IIS, Zeus) •NSAPI (Netscape iPlanet) •Java servlet •AOLServer •Roxen <p>Para dar una idea del futuro que tiene PHP en el mundo de internet, a continuación se muestra un listado de los servidores que próximamente se espera que sean compatibles con PHP:</p> <ul style="list-style-type: none"> •Apache 2.0 •WSAPI (O'Reilly WebSite) •phttpd •thttpd
Windows NT, Windows 2000 Winme	<ul style="list-style-type: none"> •UNIX (todas las variantes) •Win32 (NT/W95/W98/W2000) •QNX •Mac (WebTen) •OS/2 •BeOS <p>Próximamente estará disponible en los servidores:</p> <ul style="list-style-type: none"> •OS/390 •AS/400

Bases de datos soportadas con PHP

- Adabas D
- Empress
- IBM DB2
- Informix
- Ingres
- Interbase

- Frontbase
- mSQL
- Direct MS-SQL
- MySQL
- ODBC
- Oracle (OCI7,OCI8)
- PostgreSQL
- Raima Velocis
- Solid
- Sybase
- dBase
- filePro (solo lectura)
- dbm (ndbm, gdbm, Berkeley db)

Utilidades para instalar ambientes integrados con PHP

Cuando se habla de ambientes integrados en PHP, se está haciendo alusión a la posibilidad de trabajar con una serie de herramientas, además de PHP, que le ayudan al desarrollador de aplicaciones a crear sistemas multinivel capaces de ser puestos en la web y acceder a motores de bases de datos.

En internet existe la posibilidad de conseguir una herramienta superpoderosa, que la mayoría de principiantes en la programación PHP conoce. Esta maravillosa herramienta se llama “appserv”, y al instalarse en una máquina con sistema operativo Windows permite tener una serie de instrumentos para trabajar con PHP, incluyendo el servidor web y una serie de configuraciones que de otra manera hubiera tenido que realizar el programador. Estas tareas ahora se vuelven cosas fáciles, puesto que el *software* realiza todas las labores de configuración necesarias para que después de instalar dicha herramienta el lector pueda empezar a trabajar con PHP y realizar aplicaciones web con este maravilloso lenguaje. La versión 2.0 de appserv incluye los siguientes paquetes:

- Apache WebServer Version 1.3.27
- PHP Script Language Version 4.3.1
- MySQL Database Version 4.0.12
- PHP-Nuke Web Portal System Version 6.5
- phpMyAdmin Database Manager Version 2.4.0

Cómo crear clases con PHP

Como hemos dicho, es posible crear clases con PHP. A continuación se muestra una página sencilla hecha en PHP, que crea una clase denominada “figura”, la cual tiene tres atributos y un constructor, que es el encargado de imprimir en la página los valores de dichos atributos. Veamos el código:

```
<?php
class figura{
    var $area;
    var $perimetro;
    var $lados;

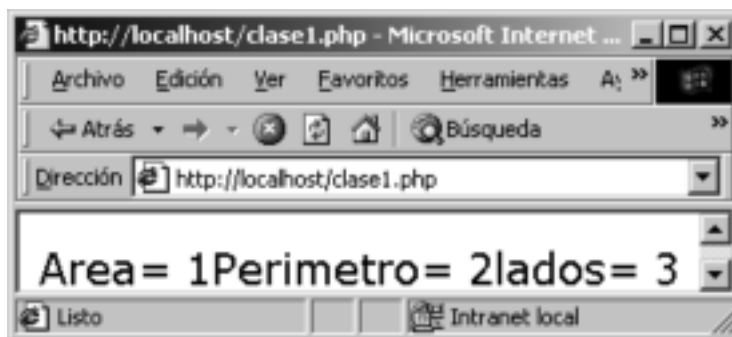
    function figura(){
        $area=1;
        $perimetro=2;
        $lados=3;
        echo 'Area= ' . $area;
        echo 'Perimetro= ' . $perimetro;
        echo 'lados= ' . $lados;
    }

}

$objeto1 = new figura();

?>
```

El resultado de la ejecución de la anterior página es similar al siguiente:



Error común al inicializar variables en clases con PHP

Un error muy común al inicializar las variables contenidas en una clase es inicializarlas cuando se declaran. Veamos como ejemplo el siguiente código:

```
<?php
class figura{
    var $area=1;
    var $perimetro=2;
    var $lados=3;

    function figura(){
        echo 'Area= ' . $area;
        echo 'Perimetro= ' . $perimetro;
        echo 'lados= ' . $lados;
    }

}

$objeto1 = new figura();

?>
```

Produce un resultado similar al siguiente:



Con lo cual el lector podrá apreciar que de nada sirvió asignarle valores a las variables en la misma declaración de estas, debido a que el constructor no las toma, y para ser más exactos, ninguna función reconoce los valores que se hayan asignado en la misma declaración global de las variables. Para corregir este problema es necesario inicializar las variables en el constructor, tal como se muestra en el siguiente código:

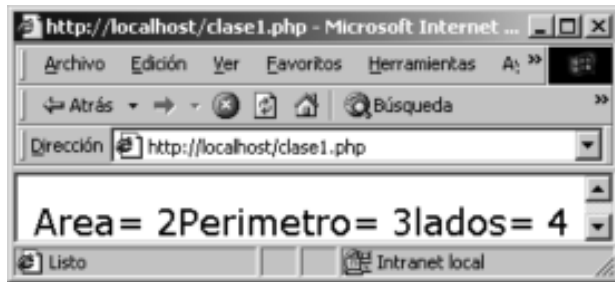
```
<?php
class figura{
    var $area;
    var $perimetro;
    var $lados;

    function figura(){
        $area=2;
        $perimetro=3;
        $lados=4;
        echo 'Area= ' . $area;
        echo 'Perimetro= ' . $perimetro;
        echo 'lados= ' . $lados;
    }

}

$objeto1 = new figura();
?>
```

El cual arrojará un resultado similar al siguiente:



Es importante que el lector comprenda que la función “figura()” es una función constructora debido a que tiene el mismo nombre de la clase, pues para este caso en particular, la clase que estamos trabajando tiene el nombre “figura”.

Al crear un objeto con la instrucción:

```
$objeto1 = new figura();
```

estamos creando un objeto que tiene como plantilla, por decirlo de alguna manera, a la clase “figura”, dado que esta tiene un constructor que se invoca y que asigna valores a las variables globales de la clase “figura” e imprime los valores de dichas variables tal como muestra el siguiente código:

```

function figura(){
    $area=2;
    $perimetro=3;
    $lados=4;
    echo 'Area= ' . $area;
    echo 'Perimetro= ' . $perimetro;
    echo 'lados= ' . $lados;
}
}

```

El polimorfismo en los constructores de las clases

En algunas ocasiones se dice que los lenguajes orientados a objetos aceptan polimorfismo. Este concepto aplicado a la función que actúa como constructora de una clase quiere decir que cuando se crea un objeto, el servidor decide qué constructor utilizar. Por ejemplo, veamos el siguiente código:

```

<?php
class figura{
    var $area;
    var $perimetro;
    var $lados;

    function figura(){
        $area=2;
        $perimetro=3;
        $lados=4;
        echo 'Area= ' . $area;
        echo 'Perimetro= ' . $perimetro;
        echo 'lados= ' . $lados;
    }

    function figura($parametro1){
        echo 'Este constructor no hace nada';
    }
}

```

```
$objeto1 = new figura();  
$objeto2 = new figura("LUIS FELIPE WANUMEN SILVA");  
  
?>
```

Este código hará que el lector piense que la creación del segundo objeto obliga a que se llame a la función “figura(\$parametro1)”, y la verdad es que esto sucede así en algunas versiones de PHP, lo cual es importante tener en cuenta.

En forma personal, le puedo contar al lector que probé esta función sin éxito alguno. Según lo que he trabajado, en algunas ocasiones me funcionó cuando utilizaba otra versión de PHP, pero puedo asegurar que este código no sirve en todas las versiones. Por esta razón aconsejo, en la medida de lo posible, no utilizar todavía este tipo de polimorfismo, puesto que no se puede asegurar que en todos los servidores va a funcionar.

Funciones en clases y operador this

Antes de mostrar el ejercicio es bueno explicar que para crear funciones en una clase en PHP es necesario colocar la palabra “function” antes del nombre de la función. Por lo tanto, la sintaxis para crear funciones será:

```
function nombre_de_la_función(){  
  
}
```

En el caso de funciones que reciban parámetros se utiliza la siguiente sintaxis:

```
function nombre_de_la_función($nombre_del_parámetro){  
  
}
```

Por otra parte es bueno anotar que el operador this sirve para acceder a una variable o a un método de una clase. Veamos por ejemplo el siguiente código:

```
function nombre_de_la_función($nombre_del_parámetro){  
    $this->variable1 = 10;  
}
```

Asume que en la clase a la que pertenece la función existe una variable denominada “variable1” y se le asigna el valor de “10”.

Dado que viendo código es como se aprende, a continuación se muestra el código de una página, el cual luego se explica en detalle:


```

<HTML>
<BODY>
<?php
class figura{
var $area;
var $perimetro;
var $lados;

function figura(){
$area=2;
$perimetro=3;
$lados=4;
echo 'Area= ' . $area;
echo '<BR> ';
echo 'Perimetro= ' . $perimetro;
echo '<BR> ';
echo 'lados= ' . $lados;
}

function area($PAR){
echo "Llamada a area";
$this->area = $PAR;
echo '<BR> ';
echo 'Area= ' . $this->area;
}
}
?>

<TABLE border=2>
<TR>
<TD>
Creando primer objeto
</TD>
<TD>
<?php
$objeto1 = new figura();
?>
</TD>

```

```

</TR>
<TR>
<TD>
Creando segundo objeto
</TD>
<TD>
<?php
$objeto2 = new figura();
?>
</TD>
<TD>
</TR>
<TR>
<TD>
Llamada a area() del objeto2
</TD>
<TD>
<?php
$objeto2->area(10);
?>
</TD>
</TR>
</TABLE>
</BODY>
</HTML>

```

Este código crea un objeto referenciado con la variable “\$objeto1”, el cual es de tipo “figura”. Al crear a dicho objeto se llama inmediatamente al constructor, que inicializa las variables globales de la clase e imprime sus valores.

Recordemos que la instrucción

```
echo '<BR> ';
```

imprime un texto en la página. En este caso se están imprimiendo unos caracteres que en HTML significan un salto de carro, y dado que las instrucciones están en una página HTML, se produce dicho efecto.

Los resultados de la creación del primer objeto son:

Área= 2 Perímetro= 3 Lados= 4

Pero ya que dicha creación de objeto se encuentra dentro de la fila de una tabla, los resultados serán parciales, como se ve a continuación:

Creando primer objeto	Área= 2 Perímetro= 3 Lados= 4
-----------------------	-------------------------------------

La instrucción

```
$objeto2 = new figura();
```

similar al caso explicado con la variable “\$objeto1”, crea un objeto denominado “\$objeto2”, el cual también es una instancia de la clase “figura” e imprime los contenidos de sus variables.

Con esto, los resultados parciales de la creación del segundo objeto serán los siguientes:

Creando segundo objeto	Área= 2 Perímetro= 3 Lados= 4
------------------------	-------------------------------------

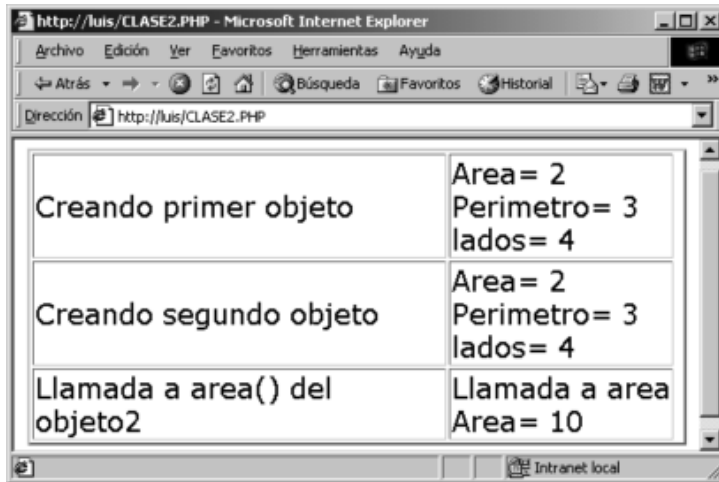
Lo interesante viene al ejecutar en el servidor la instrucción:

```
$objeto2->area(10);
```

pues en este momento se imprime

Llamada a área Área= 10

Con el ánimo de dar una visión completa al lector sobre los resultados de la ejecución de la página, veamos el resultado:



La herencia en PHP

La herencia en PHP es una de las cosas más interesantes que tiene este lenguaje, pues permite crear clases que hereden de unas clases bases. Al igual que con otros lenguajes, cuando se hereda de una clase, la clase hija puede utilizar los métodos y atributos de la clase padre. En otras palabras, los hijos heredan todo lo que dejan los padres, tal como en la vida real, pero al revés no funciona, es decir que no es posible acceder a un método o atributo de un padre que haya sido declarado únicamente en el hijo.

Para aclarar más este concepto de herencia, a continuación se muestra un ejercicio que declara una clase “padre”, que tiene un constructor y un método denominado “hablar()”. También se declara una clase denominada “hijo”, que también tiene su constructor y un método denominado “cantar()”.

Con la situación anterior podemos pensar que una vez creado un objeto de tipo “padre” y otro de tipo “hijo”, es posible acceder al método “hablar()” del padre y al método “cantar()” del hijo. Pero lo más interesante viene cuando intentamos llamar al método “hablar()”, puesto que a pesar de no estar directamente declarado en la clase “hijo”, vemos que sí es posible utilizar este método por cuanto se hereda de la clase “padre”, de la cual extiende la clase “hijo”.

Ahora bien, menos charla y más trabajo, veamos el código:

```
<?php

class padre{

    function hablar(){
        echo 'Soy una persona...';
    }
}
```

```

}

function padre(){
echo 'Creando instancia de padre...';
}

}

class hijo extends padre{

function cantar(){
echo 'Lunita consentida...';
}

function hijo(){
echo 'Creando instancia de hijo...';
}
}

?>
<HTML>
<BODY>
<CENTER>
<TABLE BORDER=2>

<TR>
<TD>
Creación del Padre
</TD>
<TD>
<?php
$persona1 = new padre();
?>
</TD>
</TR>

<TR>

```

```
<TD>
Creación del Hijo
</TD>
<TD>
<?php
$persona2 = new hijo();
?>
</TD>
</TR>
```

```
<TR>
<TD>
Método Hablar del Padre
</TD>
<TD>
<?php
$persona1->hablar();
?>
</TD>
</TR>
```

```
<TR>
<TD>
Método cantar del Hijo
</TD>
<TD>
<?php
$persona2->cantar();
?>
</TD>
</TR>
```

```
<TR>
<TD>
Método Hablar del Hijo
</TD>
```

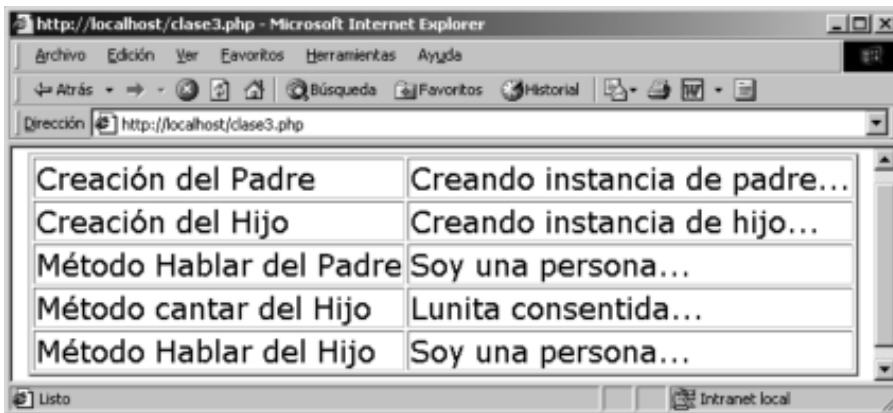
```

<TD>
<?php
$persona2->hablar();
?>
</TD>
</TR>

</TABLE>
</CENTER>
</BODY>
</HTML>

```

Es interesante observar que el método “hablar()” del objeto “\$persona2” no fue declarado explícitamente en la clase “hijo”, pero puede ser utilizado debido a que es como si estuviera presente en la clase “hijo” por ser un método definido en una clase padre. Si montamos la página anterior en un servidor que permita correr páginas PHP, tendríamos un resultado similar al siguiente:



Hay tener en cuenta que si intentamos llamar al método “cantar()” del objeto referenciado por la variable “\$persona1” tendríamos un error, y dependiendo del servidor PHP en el que estemos ejecutando la página se presentaría el error o simplemente se mostraría una página en blanco.

Sobreescribir métodos en clases heredadas

Imagínese que tenemos una clase base con ciertos atributos. Después imagínese que tenemos una clase que hereda de la primera clase y que declara también sus métodos. Es posible que existan dos métodos con el mismo nombre: uno en la clase base y otro en la clase que hereda. Surge la pregunta: ¿es bueno permitir que se presenten estas situaciones? La respuesta es muy sencilla: no solamente se permite que sucedan

estas cosas, sino que incluso a veces, en términos de programación, se recomienda que existan para especificar métodos que se especializan, pero que por la naturaleza del problema conviene que tengan el mismo nombre que algún método definido en la clase base.

En el siguiente ejercicio se presenta una clase denominada “padre”, que tiene el método “cantar()”; y una clase que hereda de esta denominada “hijo”, que tiene el método “cantar()” también definido. Es importante notar que el método “cantar()” definido en la clase hijo reemplaza al método “cantar()” especificado en la clase padre.

Veamos el código de la siguiente página:

```
clase5.php
<?php

class padre{

function cantar(){
echo 'Dejaré mi tierra por tí...';
}

function padre(){
echo 'Creando instancia de padre...';
}

}

class hijo extends padre{

function cantar(){
echo 'Lunita consentida...';
}

function hijo(){
echo 'Creando instancia de hijo...';
}
}

?>
<HTML>
```



```

<BODY>
<CENTER>
<TABLE BORDER=2>

<TR>
<TD>
Creación del Padre
</TD>
<TD>
<?php
$persona1 = new padre();
?>
</TD>
</TR>

<TR>
<TD>
Creación del Hijo
</TD>
<TD>
<?php
$persona2 = new hijo();
?>
</TD>
</TR>

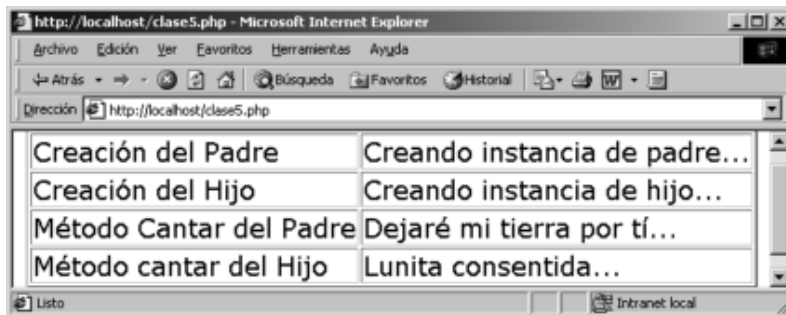
<TR>
<TD>
Método Cantar del Padre
</TD>
<TD>
<?php
$persona1->cantar();
?>
</TD>
</TR>

```

```
<TR>
<TD>
Método cantar del Hijo
</TD>
<TD>
<?php
$persona2->cantar();
?>
</TD>
</TR>

</TABLE>
</CENTER>
</BODY>
</HTML>
```

Al ser ejecutada, la página genera un resultado en el explorador similar al siguiente:



En algunos lenguajes como Java se presenta la función “super()”, la cual sirve para llamar métodos de la clase padre cuando los métodos de las clases hijas han sido reemplazados por otros que tienen el mismo nombre en la clase hija. Pero por desgracia esta funcionalidad no aplica para lenguajes como PHP, en otras palabras, en el programa anterior no es posible llamar al objeto referenciado por la variable “\$persona2” para usar la función “cantar()” definida en la clase denominada “padre”.

Podemos resumir lo antes dicho, afirmando que los métodos que se definan en una clase hija y que tengan el mismo nombre de algún método definido en una clase antecesora, reemplazan absolutamente a dicho método y no es posible, por lo menos en las versiones actuales de PHP, hacer alusión a estos métodos.

Herencia múltiple en PHP

Es bueno tener en cuenta que como tal, el término herencia múltiple no aplica para el lenguaje de programación PHP, puesto que no es posible que una clase herede al mismo tiempo de una clase padre y de una clase madre. Esto no se debe confundir con que entre sus ancestros se encuentra un padre y una madre.

Veamos el siguiente código de una página que no produce errores al ser corrida por el servidor:

```
clase6.php
<?php

class abuelo{
function abuelo(){
echo 'Creando instancia de abuelo...';
}
}

class abuela{
function abuela(){
echo 'Creando instancia de abuela...';
}
}

class padre extends abuelo{
function padre(){
echo 'Creando instancia de padre...';
}
}

class madre extends abuelo{
function madre(){
echo 'Creando instancia de madre...';
}
}

class hijo extends madre{
function hijo(){
```

```
echo 'Creando instancia de hijo...';  
}  
}
```

```
class hija extends madre{  
function hija(){  
echo 'Creando instancia de hija...';  
}  
}
```

```
?>
```

```
<HTML>
```

```
<BODY>
```

```
<CENTER>
```

```
<TABLE BORDER=2>
```

```
<TR>
```

```
<TD>
```

```
Creación del Abuelo
```

```
</TD>
```

```
<TD>
```

```
<?php
```

```
$persona1 = new abuelo();
```

```
?>
```

```
</TD>
```

```
</TR>
```

```
<TR>
```

```
<TD>
```

```
Creación del Abuela
```

```
</TD>
```

```
<TD>
```

```
<?php
```

```
$persona2 = new abuela();
```

```
?>
```

```
</TD>
```

```

</TR>

<TR>
<TD>
Creación de Padre
</TD>
<TD>
<?php
$persona3 = new padre();
?>
</TD>
</TR>

<TR>
<TD>
Creación de Madre
</TD>
<TD>
<?php
$persona4 = new madre();
?>
</TD>
</TR>

<TR>
<TD>
Creación de Hijo
</TD>
<TD>
<?php
$persona5 = new hijo();
?>
</TD>
</TR>

<TR>

```

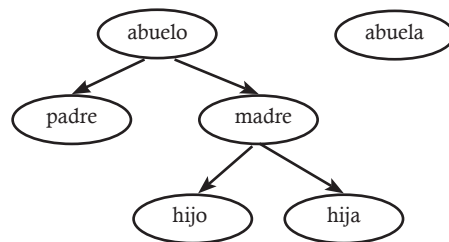
```

<TD>
Creación de Hija
</TD>
<TD>
<?php
$persona6 = new hija();
?>
</TD>
</TR>

</TABLE>
</CENTER>
</BODY>
</HTML>

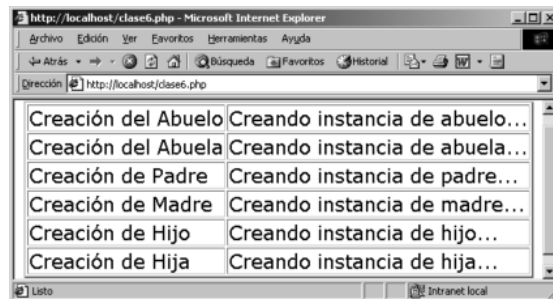
```

Esquemáticamente hablando, podemos describir la anterior secuencia de herencia de la siguiente manera:



En el gráfico anterior podemos ver que una clase tiene como antecesora inmediata máximo a otra clase y no se da el caso que tenga como antecesora inmediata a dos clases. Es bueno que el lector observe que en la herencia anterior, los antecesores, por ejemplo de la clase denominada “hijo”, son “madre” y “abuelo”, pero su grado de parentesco hacia arriba en el árbol genealógico los posiciona en grados diferentes.

A continuación observaremos el resultado de ejecutar la página anterior en un servidor web:



Con todo lo dicho, nos podemos hacer la siguiente pregunta con el ánimo de verificar que se ha comprendido que PHP no acepta herencia múltiple: ¿el siguiente código genera error?

```

clase7.php
<?php

class abuelo{
function abuelo(){
echo 'Creando instancia de abuelo...';
}
}

class abuela{
function abuela(){
echo 'Creando instancia de abuela...';
}
}

class padre extends abuelo extends abuela{
function padre(){
echo 'Creando instancia de padre...';
}
}

?>
<HTML>
<BODY>
<CENTER>
<TABLE BORDER=2>

<TR>
<TD>
Creación del Abuelo
</TD>
<TD>
<?php
$persona1 = new abuelo();
?>

```

```

</TD>
</TR>

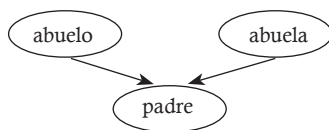
<TR>
<TD>
Creación del Abuela
</TD>
<TD>
<?php
$persona2 = new abuela();
?>
</TD>
</TR>

<TR>
<TD>
Creación de Padre
</TD>
<TD>
<?php
$persona3 = new padre();
?>
</TD>
</TR>

</TABLE>
</CENTER>
</BODY>
</HTML>

```

A lo que responderemos que sí, puesto que esquemáticamente hablando el anterior código implementa la siguiente herencia:



Como hemos visto, esto no está permitido en PHP, por lo menos en las versiones actuales. Algunos dicen que en próximas versiones se podrá implementar la herencia múltiple en PHP. Esperamos que así sea.

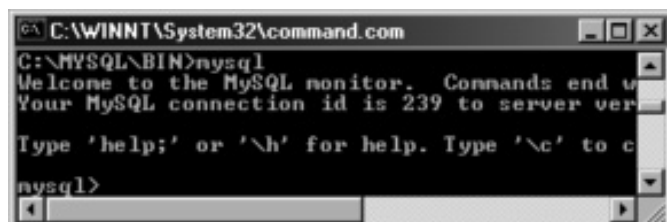
Desarrollar aplicaciones PHP con bases de datos

El servidor de bases de datos MySQL

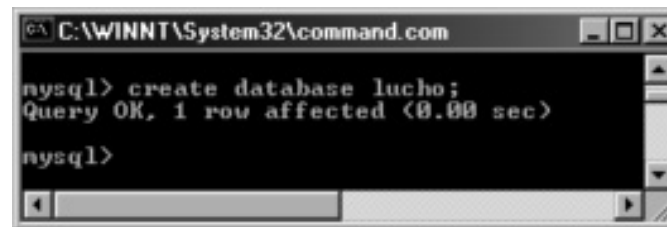
Una vez que ha instalado MySQL, puede iniciar el servidor con la instrucción “MySQL”. Este comando será visible si usted está ubicado en el directorio donde se encuentra el ejecutable de MySQL. Veamos la figura ilustrativa:



El lector puede darse cuenta que el servidor ha arrancado porque le aparece una pantalla similar a la siguiente:



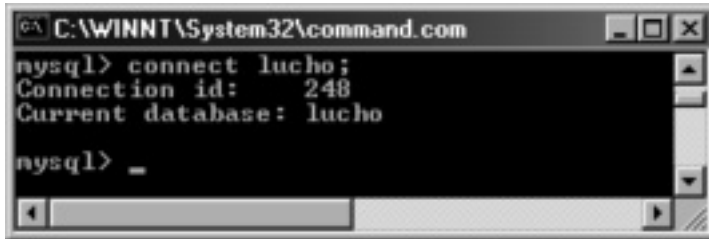
El usuario de la base de datos, o quien tenga permisos para crear una base de datos en el servidor, lo puede hacer de la siguiente manera:



Luego se puede conectar con esta base de datos, mediante la instrucción

```
Connect lucho
```

tal como se puede apreciar en la figura:

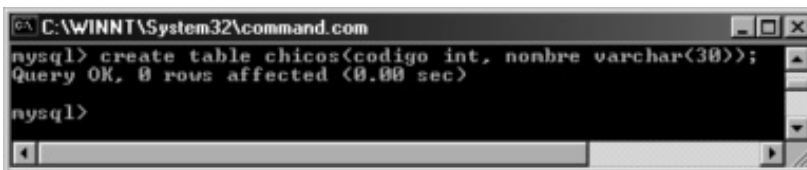


```
mysql> connect lucho;
Connection id: 248
Current database: lucho
mysql> _
```

Una vez creada la base de datos y conectados a ella, podemos hacer tablas. Para ello, utilizaremos la siguiente instrucción:

```
Create table chicos(codigo int, nombre varchar(30));
```

La ejecución de la anterior instrucción en la base de datos MySQL se puede observar en la siguiente figura:



```
mysql> create table chicos(codigo int, nombre varchar(30));
Query OK, 0 rows affected (0.00 sec)
mysql>
```

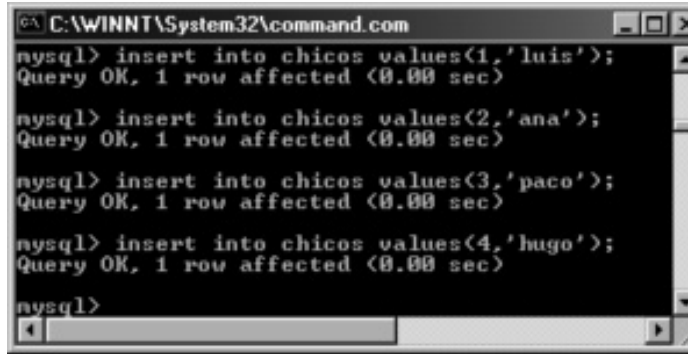
Si usted es un poco incrédulo, puede verificar que la tabla fue creada. Mediante la siguiente sentencia se dará cuenta de que esto es así:



```
mysql> select * from chicos;
Empty set (0.00 sec)
mysql>
```

Con esto, el servidor de bases de datos MySQL, le está diciendo que la tabla “chicos” está vacía y por el momento no tiene registros.

Ahora bien, con el ánimo de lograr tener algunos datos de prueba, a continuación se hacen algunas inserciones sobre la tabla. Veamos:



```

C:\WINNT\System32\command.com
mysql> insert into chicos values(1,'luis');
Query OK, 1 row affected (0.00 sec)

mysql> insert into chicos values(2,'ana');
Query OK, 1 row affected (0.00 sec)

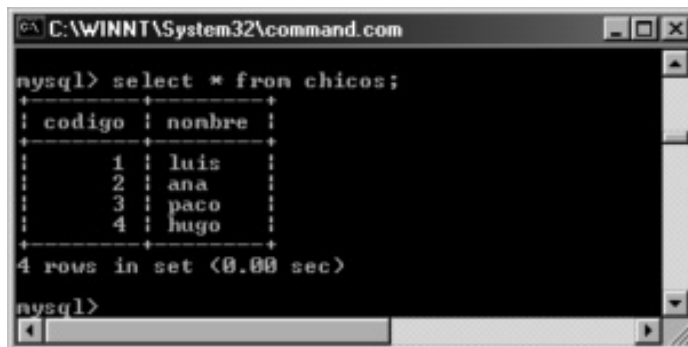
mysql> insert into chicos values(3,'paco');
Query OK, 1 row affected (0.00 sec)

mysql> insert into chicos values(4,'hugo');
Query OK, 1 row affected (0.00 sec)

mysql>

```

Una vez insertados los registros en el servidor de bases de datos MySQL, podemos verificar que se han agregado dichos valores haciendo una selección de todos los registros de la tabla “chicos”, tal como se muestra a continuación:



```

C:\WINNT\System32\command.com
mysql> select * from chicos;
+----+-----+
| codigo | nombre |
+----+-----+
| 1      | luis   |
| 2      | ana    |
| 3      | paco   |
| 4      | hugo   |
+----+-----+
4 rows in set (0.00 sec)

mysql>

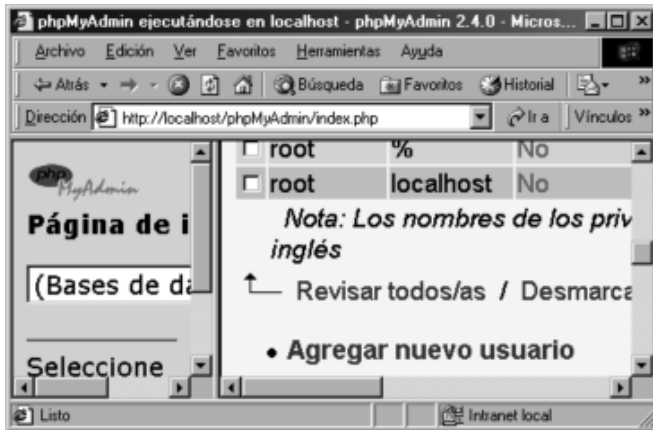
```

Algo importante a tener en cuenta al hacer aplicaciones con motores de bases de datos, es que por lo general estos tienen una forma de administración mediante usuarios, con el ánimo de dar niveles de acceso a las distintas personas que quieran ingresar a las bases de datos del servidor o que quieran realizar operaciones sobre las tablas de las bases de datos contenidas en el servidor.

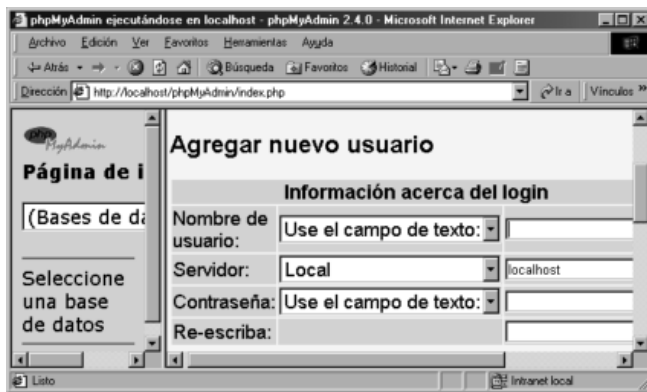
Por todo lo mencionado, podemos decir que en el caso de MySQL se puede hacer uso de la herramienta web que trae incorporada con el ánimo de crear los usuarios.



En dicha herramienta escogemos la opción “Privilegios”, con lo cual aparecen los diversos usuarios que están incluidos en el gestor de bases de datos. Allí encontramos una opción que nos permite crear un nuevo usuario, tal como se muestra en la siguiente figura:

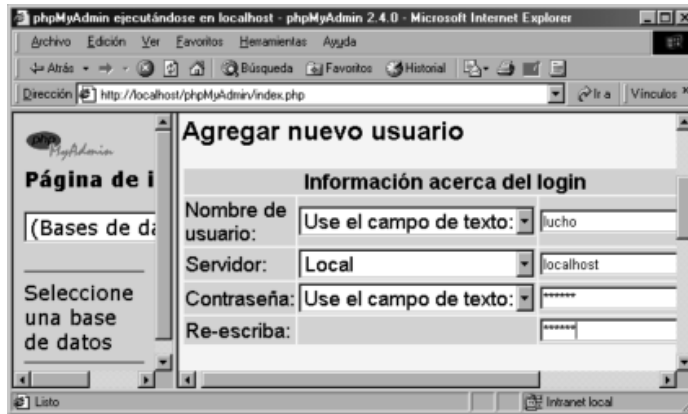


Allí podemos apreciar una interfaz similar a la siguiente



en la cual se digita el nombre del usuario, el servidor al que va a tener derecho de acceso este usuario y la contraseña de registro.

Para el caso de nuestro ejercicio, vamos a escoger los valores mostrados a continuación:



En la parte de abajo de la página anterior se pueden establecer los privilegios globales para el usuario “lucho”. Veamos:



En este caso, dado que se trata de un ejercicio académico, se le dieron todos los permisos al usuario “lucho” y se le digitó como *password* el valor “123456”.

La conexión con el servidor de bases de datos MySQL

En el siguiente código se puede apreciar que en PHP es posible crear una variable y, sin definirle el tipo, asignarle el resultado de una conexión. Es bueno recordar que en PHP los nombres de las variables están precedidas por el símbolo: “\$”

```
<html>
<body>

<?php
```

```
if(!($conexion=mysql_connect("localhost","lucho","123456")))  
{  
    echo "Errores conectando con el servidor";  
    exit();  
}  
else  
{  
    echo "Conexion correcta".<br>;  
}  
  
?>  
  
</body>  
</html>
```

La ejecución de la página anterior produce un resultado similar al siguiente:



La conexión con la base de datos ubicada en el servidor de bases de datos

```
<html>  
<body>  
  
<?php  
if(!($conexion=mysql_connect("localhost","lucho","123456")))
```

```

{
echo "Errores conectando con el servidor";
exit();
}
else
{
echo "Conexion correcta."<br>";
}

if (!mysql_select_db("lucho",$conexion))
{
echo "Error conectando con la base de datos.";
echo "<br>";
exit();
}
else{
echo "Base de datos correcta";
echo "<br>";
}

?>

</body>
</html>

```

La ejecución de la página anterior en el explorador web nos produce un resultado similar al siguiente:



La conexión con tablas de la base de datos

```
<html>
<body>

<?php
if(!($conexion=mysql_connect("localhost","lucho","123456")))
{
echo "Errores conectando con el servidor";
exit();
}
else
{
echo "Conexion correcta"."<br>";
}

if (!mysql_select_db("lucho",$conexion))
{
echo "Error conectando con la base de datos.";
echo "<br>";
exit();
}
else{
echo "Base de datos correcta";
echo "<br>";
}

$result=mysql_query("select * from chicos",$conexion);

mysql_free_result($result);
mysql_close($link);
?>

</body>
</html>
```

La ejecución de la página anterior en el explorador web nos produce un resultado similar al siguiente:



Mostrar registros de una tabla en MySQL desde PHP versión 1

```
<html>
<body>

<?php
if(!($conexion=mysql_connect("localhost","lucho","123456")))
{
    echo "Errores conectando con el servidor";
    exit();
}
else
{
    echo "Conexion correcta."<br>";
}

if (!mysql_select_db("lucho",$conexion))
{
    echo "Error conectando con la base de datos.";
    echo "<br>";
    exit();
}
else{
    echo "Base de datos correcta";
    echo "<br>";
}
```

```
$result=mysql_query("select * from chicos",$conexion);

echo "<table>";
while($row = mysql_fetch_array($result)) {
    printf("<tr><td>%s</td><td>%s</td></tr>", $row["codigo"],$row["nomb
re"]);
}
echo "</table>";
mysql_free_result($result);
mysql_close($link);
?>

</body>
</html>
```

La ejecución de la página anterior nos muestra un resultado similar al siguiente:



Mostrar registros de una tabla en MySQL desde PHP versión 2

El ejercicio anterior se puede desarrollar con una pequeña modificación en la forma como se enseñan los resultados. En esta ocasión se muestran los datos con la función “mysql_result” con el ánimo de explicar el funcionamiento de dicha función:

```
<html>
<body>

<?php

if (!($conexion=mysql_connect("localhost","lucho","123456")))
{
```

```

echo "Errores conectando con el servidor";
exit();
}
else{
echo "Conexion correcta."<br>";
}

if (!mysql_select_db("lucho",$conexion))
{
echo "Error conectando con la base de datos.";
echo "<br>";
exit();
}
else{
echo "Base de datos correcta";
echo "<br>";
}

$result=mysql_query("select * from chicos",$conexion);

echo "<table>";
$cont = 0;
while($row = mysql_fetch_array($result)) {
echo "<tr>";
echo "<td>";
echo "Código: ".mysql_result($result, $cont, "codigo")."<br>";
echo "</td>";
echo "<td>";
echo "Nombre: ".mysql_result($result, $cont, "nombre")."<br>";
echo "</td>";
echo "<tr>";
$cont= $cont +1;
}
echo "</table>";
mysql_free_result($result);
mysql_close($link);

```

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

Consultar un registro en una base de datos MySQL

```
<html>
<head>
<title>Animo lunita</title>
</head>

<body>
<form ACTION="login1.php">
  <TABLE>

    <tr>
<td>Usuario:</td>
<td> <input name="usuario" size="18" value= ""> </td>
    </tr>

    <tr>
<td>Usuario:</td>
<td> <input name="clave" size="18" value= ""> </td>
    </tr>

    <TR>
    <TD> <input type=RESET name=BOTON2 value ="CANCELAR" >
</TD>
    <TD> <input type=SUBMIT name= BOTON1 value= "INGRESAR" > </
TD>
    </TR>

  </table>

</form>

<?php
if($usuario!="" && $clave!="")
```

```

{
$user="root";
$password="";
$bd="lucho";
$host="localhost";

if(!($conexion=mysql_connect($host,$user,$password)))
{
echo "Errores conectando con el servidor";
exit();
}
else
{
echo "Conexion correcta.".<br>";
}

if (!mysql_select_db("lucho",$conexion))
{
echo "Error conectando con la base de datos.";
echo "<br>";
exit();
}
else{
echo "Base de datos correcta";
echo "<br>";
$sql = "select * from usuario where nombre=".$usuario." and clave=".$clave.".";
printf("%s", $sql);

$result = mysql_query($sql,$conexion);
echo "<table>";
while($row = mysql_fetch_array($result)) {
printf("<tr><td>%s</td><td>%s</td></tr>",
$row["nombre"],$row["clave"]);
}
}

```

```
echo "</table>";
mysql_free_result($result);
mysql_close($conexion);
}
}

?>

</body>
</html>

</body>
</html>
```

Hacer búsquedas y mostrar los resultados paginados

En muchas ocasiones es necesario hacer páginas PHP en las que los usuarios puedan realizar búsquedas. El gran problema en internet es que si el resultado de la búsqueda hecha por el usuario es muy grande, es posible que la conexión se caiga o que se presenten diversos problemas relacionados con la demora en la conexión. Por esta razón es importante tener presente que existen métodos para paginar los resultados, es decir que podemos hacer uso de la paginación como un recurso de programación que permite la disminución de recursos de máquina y de red para realizar procesos de enviar grandes cantidades de datos a los clientes de internet, desde un servidor de bases de datos, pasando obviamente por un servidor web.

Supongamos inicialmente la siguiente página PHP:

```
<html>
<head>
    <title>Paginación con PHP</title>
</head>

<body>
<form action="buscador.php" method="get">
<input type="text" name="criterio" size="22" maxlength="150">
<input type="submit" value="Buscar">
</form>
<?
if($criterio!="")
{
```

```

//conecto con la base de datos
$conn = mysql_connect("luis","seminario","123456789");
mysql_select_db("mysql",$conn);

//inicializo el criterio y recibo cualquier cadena que se desee buscar
//$criterio = "";
if ($_GET["criterio"]!="")
{
    $txt_criterio = $_GET["criterio"];
    $criterio = " where nombre like '%" . $txt_criterio . "%'";
} // $sql = "SELECT * FROM estudiante WHERE ". $campo . " LIKE '%" .
$valor . "%'";

//Limito la busqueda
$TAMANO_PAGINA = 3;

//examino la página a mostrar y el inicio del registro a mostrar
$pagina = $_GET["pagina"];
if (!$pagina) {
    $inicio = 0;
    $pagina=1;
}
else {
    $inicio = ($pagina - 1) * $TAMANO_PAGINA;
}

//miro a ver el número total de campos que hay en la tabla con esa búsqueda
$$sql = "select * from estudiante " . $criterio;
$rs = mysql_query($sql,$conn);
$num_total_registros = mysql_num_rows($rs);
//calculo el total de páginas
$total_paginas = ceil($num_total_registros / $TAMANO_PAGINA);

//pongo el número de registros total, el tamaño de página y la página que se
muestra
echo "Número de registros encontrados: " . $num_total_registros . "<br>";
echo "Se muestran páginas de " . $TAMANO_PAGINA . " registros cada
una<br>";

```

```

echo "Mostrando la página " . $pagina . " de " . $total_paginas . "<p>";

//construyo la sentencia SQL
$ssql = "select * from estudiante " . $criterio . " limit " . $inicio . "," . $TAMANO_PAGINA;
//echo $ssql . "<p>";

$rs = mysql_query($ssql);

echo "<table border=\"1\">
<tbody>
<tr>
<td>Codigo</td>
<td>Nombre</td>
</tr>";
while ($row=mysql_fetch_array($rs))
{
    Printf("<tr> <td> %s</td><td>%s</td> </tr>", $row[codigo], $row[nom
bre]);
}
}
echo "</tbody>";
echo "</table>";
?>
<?
//cerramos el conjunto de resultados y la conexión con la base de datos
mysql_free_result($rs);
mysql_close($conn);

echo "<p>";

//muestro los distintos índices de las páginas, si es que hay varias páginas
if ($total_paginas > 1){
    for ($i=1; $i<=$total_paginas; $i++)
    {
        if ($pagina == $i)
            //si muestro el índice de la página actual, no coloco enlace
            echo $pagina . " ";
    }
}

```



```

        else
            //si el índice no corresponde con la página mostrada ac-
tualmente, coloco el enlace para ir a esa página
            echo "<a href='buscador.php?pagina=" . $i . "&criterio="
. $txt_criterio . ">" . $i . "</a> ";
        }
    }
?>

</body>
</html>

```

Por el momento, no se vaya a preocupar si no entiende gran parte del código expuesto en esta página, pues se irá explicando a medida que sea necesario, con el ánimo de lograr la mayor comprensión.

Empezaremos diciendo que el cliente no ve todo lo que es código PHP, por lo tanto se tiene la seguridad de que si el usuario abre dicha página con su navegador y luego va a la opción de ver el código fuente, encontrará lo siguiente:

```

<html>
<head>
    <title>Paginación con PHP</title>
</head>

<body>
<form action="buscador.php" method="get">
<input type="text" name="criterio" size="22" maxlength="150">
<input type="submit" value="Buscar">
</form>
</tbody></table>

<p>
</body>
</html>

```

lo cual tiene una apariencia similar a



Observemos que por el código que tiene esta página, cuando se hace clic sobre el botón que tiene la etiqueta “Buscar”, en realidad estamos enviando una solicitud al servidor web para que nos procese la página “buscador.php”. Esto es posible debido a que la página “buscador.php” es recursiva, es decir, permite que se cargue una primera vez sin parámetros. En este caso tendrá un aspecto similar al anteriormente expuesto, pero también puede ser llamada mediante los dos parámetros: “página” y “criterio”.

Pero para no ir tan lejos con explicaciones, primero conectémonos al motor de la base de datos:

```
C:\WINNT\System32\command.com
Microsoft(R) Windows DOS
(C)Copyright Microsoft Corp 1998-1999.

C:\>cd appserv
C:\APPSERV>cd mysql
C:\APPSERV\MYSQL>cd bin
C:\APPSERV\MYSQL\BIN>mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4 to server version: 4.0.12-nt
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> connect mysql
Connection id: 5
Current database: mysql
mysql> _
```

Con el ánimo de ver los datos existentes en la base de datos



```

mysql> connect mysql
Connection id: 5
Current database: mysql

mysql> select * from estudiante;
+----+-----+
| codigo | nombre |
+----+-----+
| 1 | Luis |
| 2 | Carlos |
| 3 | Ana |
| 4 | Pedro |
| 5 | Sonia |
| 6 | Vanegas Carlos |
| 7 | Hector Fuquene |
| 8 | Sonia Pinzon |
| 9 | Rocio Rodriguez |
| 10 | Jorge Rodriguez |
| 12 | Jairo Ruiz |
| 11 | Wilman Navarro |
+----+-----+
12 rows in set (0.00 sec)

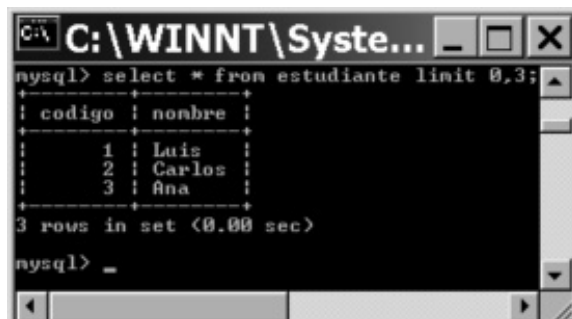
mysql>

```

y ejecutemos la sentencia

```
select * from estudiante limit 0,3
```

la cual producirá el siguiente resultado:



```

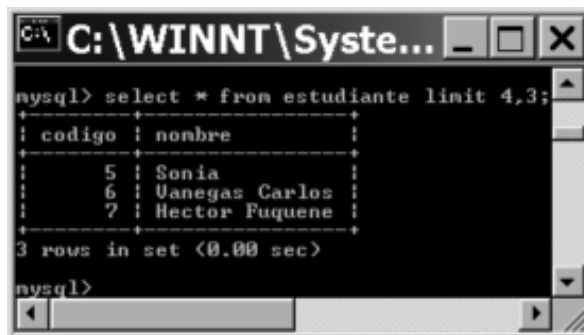
mysql> select * from estudiante limit 0,3;
+----+-----+
| codigo | nombre |
+----+-----+
| 1 | Luis |
| 2 | Carlos |
| 3 | Ana |
+----+-----+
3 rows in set (0.00 sec)

mysql>

```

Ejecutemos también la siguiente sentencia

```
select * from estudiante limit 4,3
```



```

mysql> select * from estudiante limit 4,3;
+----+-----+
| codigo | nombre |
+----+-----+
| 5 | Sonia |
| 6 | Vanegas Carlos |
| 7 | Hector Fuquene |
+----+-----+
3 rows in set (0.00 sec)

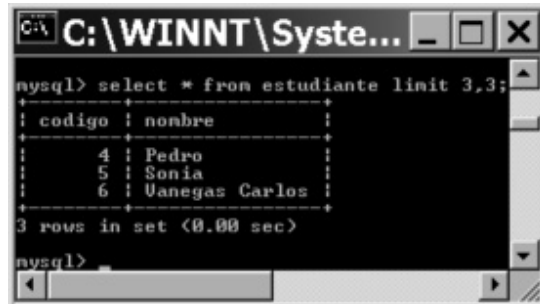
mysql>

```

Ejecutemos la sentencia

```
select * from estudiante limit 3,3
```

y obtendremos el resultado:



```
C:\WINNT\System32>mysql> select * from estudiante limit 3,3;
+----+-----+
| codigo | nombre      |
+----+-----+
| 4      | Pedro       |
| 5      | Sonia       |
| 6      | Vanegas Carlos |
+----+-----+
3 rows in set (0.00 sec)

mysql>
```

Podemos suponer que si ejecutamos la sentencia

```
select * from estudiante limit 6,3
```



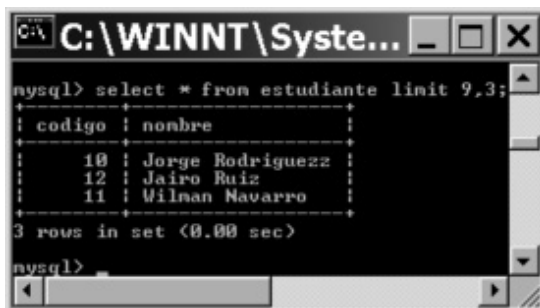
```
C:\WINNT\System32>mysql> select * from estudiante limit 6,3;
+----+-----+
| codigo | nombre      |
+----+-----+
| 7      | Hector Fuquene |
| 8      | Sonia Pinzon  |
| 9      | Rocio Rodriguez |
+----+-----+
3 rows in set (0.00 sec)

mysql>
```

y si ejecutamos la sentencia

```
select * from estudiante limit 9,3
```

obtendremos

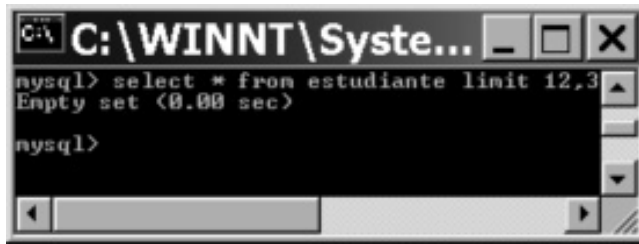


```
C:\WINNT\System32>mysql> select * from estudiante limit 9,3;
+----+-----+
| codigo | nombre      |
+----+-----+
| 10     | Jorge Rodriguez |
| 12     | Jairo Ruiz     |
| 11     | Wilnan Navarro  |
+----+-----+
3 rows in set (0.00 sec)

mysql>
```

Finalmente, y para no alargar tanto la explicación, ejecutemos la sentencia

```
Select * from estudiante limit 12,3
```



Como es posible apreciar, todas estas sentencias se pueden ejecutar creando dinámicamente el string de la sentencia, mediante la instrucción:

```
$ssql = "select * from estudiante " . $criterio . " limit " . $inicio . "," .
```

donde la variable de inicio la primera vez será cero, la segunda vez será tres, la tercera será seis, etc.

Con todas las sentencias anteriores, el lector pudo haber comprendido que esto es importante a la hora de paginar resultados y mostrarlos de esta forma en diversas páginas web.

El número de registros obtenidos como resultado de la consulta se debe dividir entre el número de registros que se quiere mostrar por página, lo que dará un número igual a la cantidad de páginas que es necesario mostrarle al navegante del sitio.

Pero ojo, hay que tener en cuenta que el número de páginas multiplicado por el número de registros mostrados por páginas no necesariamente nos da como resultado el número total de registros que son resultado de la consulta. Esto obviamente va a coincidir cuando el número de registros que son el resultado de la consulta sea un múltiplo exacto del número de páginas mostradas al navegante.

Lo anterior nos aclara la razón por la cual es necesario hacer una división entre el número total de registros y el tamaño de la página. Si el resultado es exacto, se toma tal como está, pero si el resultado tiene números del punto decimal, se procede a realizar una aproximación hacia arriba, lo que se hace mediante la instrucción

```
$total_paginas = ceil($num_total_registros / $TAMANO_PAGINA);
```

Por ejemplo, si el usuario busca los estudiantes que tenga la letra "a", el resultado será el siguiente:

En la primera página se mostrará:

Código	Nombre
2	Carlos
3	Ana
5	Sonia

En la segunda página se mostrará:

Código	Nombre
6	Vanegas Carlos
8	Sonia Pinzon
12	Jairo Ruiz

Y en la última y tercera página se mostrará:

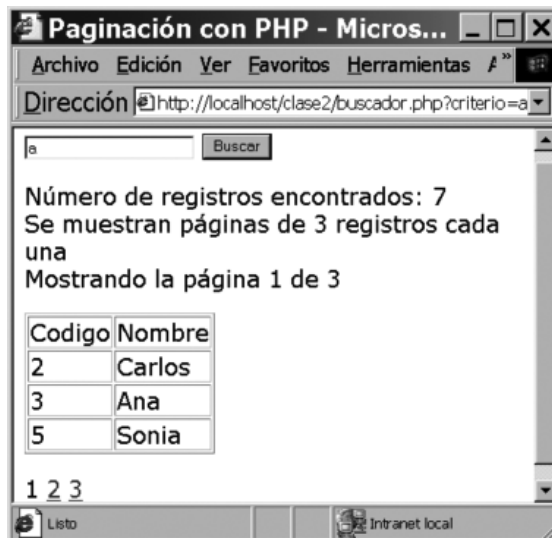
Código	Nombre
11	Wilman Navarro

Por otra parte, es interesante que el lector se dé cuenta de que la variable que guarda el número de registros que se van a mostrar por página, para este caso, es estática y precisada directamente por la página, pero en aplicaciones reales muy seguramente debe ser una variable digitada por el usuario.

En este ejemplo, dicha variable es estática y es definida mediante la instrucción:

```
$TAMANO_PAGINA = 3;
```

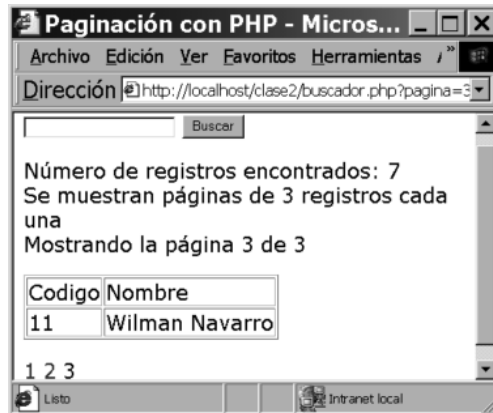
El resultado de la búsqueda será gráficamente el siguiente:



Si el navegante hace clic en el segundo enlace se mostrarán los siguientes resultados:



Y finalmente, si el usuario hace clic sobre el tercer enlace se mostrarán los siguientes resultados:



Aquí es bueno notar que cuando el número de página coincide con la variable “i”, no se coloca un hiperenlace. Esto se valida con la instrucción

```
if ($pagina == $i)
    //si muestro el índice de la página actual, no coloco enlace
    echo $pagina . “ “;
else
```

Cuando el número de la página que se está mostrando es diferente al número que se está imprimiendo en la parte inferior, es necesario hacer el hiperenlace. Esto se valida con la instrucción

```
        else
            //si el índice no corresponde con la página mostrada ac-
tualmente, coloco el enlace para ir a esa página
            echo "<a href='buscador.php?pagina=' . $i . '&criterio='
. $txt_criterio . '>' . $i . '</a> ";
        }
    }
```

Inserción de registros con PHP y MySQL

Una buena cuestión a la hora de estudiar un lenguaje que interactúa con motores de bases de datos, es conocer cómo se pueden hacer inserciones, actualizaciones y eliminaciones de registros.

En esta sección se muestra una página HTML cuyo código es el siguiente:

```
<html>
<head>
<title>Inserción de registros</title>
</head>
<body>

<table>
<form action="insertar.php" method="POST">

<tr>
<td>
Nombre:
</td>
<td>
<input type="text" name="nombre" maxlength=50>
</td>
</tr>

<tr>
<td>
E-mail:
</td>
<td>
<input type="text" name="email" maxlength=50>
</td>
</tr>
```



```
</tr>

<tr>
<td>
Direccion:
</td>
<td>
<input type="text" name="direccion" maxlength=50>
</td>
</tr>

<tr>
<td>
Ciudad:
</td>
<td>
<input type="text" name="ciudad" maxlength=50>
</td>
</tr>

<tr>
<td>
Pais:
</td>
<td>
<input type="text" name="pais" maxlength=50>
</td>
</tr>

<tr>
<td>
Cedula:
</td>
<td>
<input type="text" name="cedula" maxlength=50>
</td>
</tr>
```

```
<tr>
<td>
<input type="submit" name="enviar" value="Registrar">
</td>
<td>
<input type="reset" name="borrar" value="Borrar">
</td>
</tr>

</form>
</table>
</body>
</html>
```

La cual produce una página similar a la siguiente:



Cuando el usuario presiona el botón “Borrar”, las cajas de texto se borran y la página no hace nada más, pero cuando el usuario hace clic en el botón “Registrar”, el servidor inmediatamente pasa a la página especificada en la propiedad action del formulario, es decir que pasa a la página “insertar.php”, cuyo código fuente es el siguiente:

```
<html>
<head>
<title>Ingreso de registros</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<meta name="GENERATOR" content="Quanta Plus">
```

```

</head>
<body>

<?php

$user="seminario";
$password="123456";
$db="mysql";
$host="luis";

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

if(!$link){
die("Error al conectarse");
}

if(!mysql_select_db($db, $link ))
{
die( "No se pudo seleccionar la Base de Datos.");
}

$sql = "INSERT INTO `persona` (`nombre`, `email`, `direccion`, `ciudad`,
`pais`, `cedula`) VALUES ('$nombre', '$email', '$direccion', '$ciudad', '$pais',
'$cedula');";

mysql_query($sql);
mysql_close ($link);

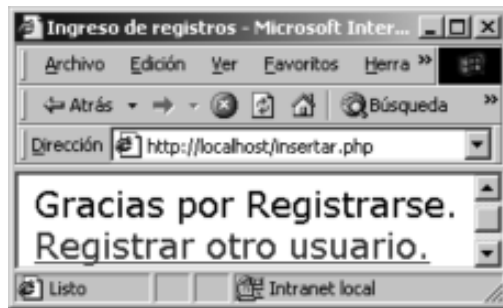
?>

Gracias por Registrarse.<br>
<a href="insertar.html">Registrar otro usuario.</a>

</body>
</html>

```

Al ser llamada desde el método action del formulario, con los datos que el usuario había digitado, la página se ve en el navegador mostrando unos resultados similares a los siguientes:



Eliminación de registros con PHP y MySQL

El siguiente ejercicio muestra una página recursiva, es decir, que se llama a sí misma cuando el usuario presiona el botón del formulario que tiene asociado el método “submit”. Veamos el código de la página:

```
<html>
<head>
<title>Registro Actualizado</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<meta name="GENERATOR" content="Quanta Plus">
</head>
<body>

<?php
if(!$cedulis){
?>
<table>
<form action="borrar.php" method="POST">

<tr>
<td>
Cedula:
</td>
<td>
<input type="text" name="cedulis" maxlength=50>
```

```

</td>
</tr>

<tr>
<td>
<input type="submit" name="enviar" value="Eliminar">
</td>
<td>
<input type="reset" name="borrar" value="Cancelar">
</td>
</tr>

</table>
<?php
}
else{
$user="seminario";
$password="123456";
$db="mysql";
$host="luis";
$link = mysql_connect ($host, $user, $password) ;
if(!$link){
die("Error al conectarse");
}
if(!mysql_select_db($db, $link ))
{
die( "No se pudo seleccionar la Base de Datos.");
}
$sql = "DELETE FROM persona WHERE cedula=".$cedula."";
echo "Registro Borrado.<br>";

mysql_query($sql);
mysql_close ($link);
}
?>
</body>
</html>

```

Para el lector que se pregunta si la variable “\$cedulis” tiene algún valor, o mejor dicho, si es posible establecer la existencia de una variable “\$cedulis”, podemos decir que la primera vez que se ejecuta la página esta variable no podrá establecerse, con lo cual se desplegará la parte concerniente a este “if” la primera vez y el resto del código que está en el “else” no se correrá. En otras palabras, la primera vez se ejecutarán las instrucciones:

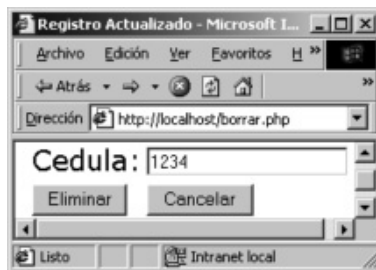
```
<table>
<form action="borrar.php" method="POST">

<tr>
<td>
Cedula:
</td>
<td>
<input type="text" name="cedulis" maxlength=50>
</td>
</tr>

<tr>
<td>
<input type="submit" name="enviar" value="Eliminar">
</td>
<td>
<input type="reset" name="borrar" value="Cancelar">
</td>
</tr>

</table>
```

Estas instrucciones muestran un formulario HTML, en el cual el usuario puede digitar el número de un usuario que desee borrar. En el navegador se verá similar a la siguiente imagen:

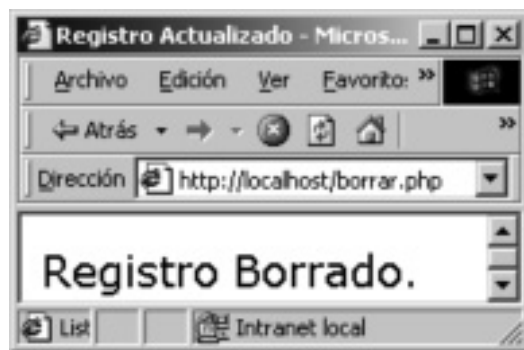


Como el lector comprenderá, cuando el usuario presione el botón “Eliminar” se ejecutará el siguiente código:

```
<?php
}
else{
$user="seminario";
$password="123456";
$db="mysql";
$host="luis";
$link = mysql_connect ($host, $user, $password) ;
if(!$link){
die("Error al conectarse");
}
if(!mysql_select_db($db, $link ))
{
die( "No se pudo seleccionar la Base de Datos.");
}
$sql = "DELETE FROM persona WHERE cedula=".$cedula."";
echo "Registro Borrado.<br>";

mysql_query($sql);
mysql_close ($link);
}
?>
```

el cual hace la conexión con la base de datos, borra el registro cuyo código ha sido ingresado por el usuario y por último cierra la conexión. A manera ilustrativa, a continuación se muestra el resultado que el usuario final puede observar después de haber presionado el botón “Eliminar”:



Parámetros y formularios con PHP_____

Los parámetros en PHP y en cualquier lenguaje de programación son importantes, pero con mayor razón en lenguajes para internet, puesto que son muchos los casos en los que se desean guardar datos que provienen de alguna página para que sean tenidos en cuenta en otra página, bien sea para dar un aspecto al usuario o para denegarle o asignarle permisos, entre otras cosas.

Por todo lo dicho, es importante que el lector comprenda que una de las formas más utilizadas en PHP para pasar parámetros es mediante formulario. Dependiendo del método utilizado en el formulario, se dice que se está pasando a otra página parámetros en forma más segura o menos segura.

En esta sección se estudian algunos casos interesantes de paso de parámetros. Se espera que el lector quede con las bases suficientes para aplicar este concepto en sus aplicaciones web de la vida real.

Pasar parámetros en formularios con el método post

Es interesante observar que si se quieren pasar parámetros de una página a otra mediante un formulario, se pueden utilizar varios métodos, entre los que figuran el método GET y el método POST. Por supuesto que existen otros métodos, pero dado que no son muy utilizados, no se describirán en este manual por ahora. Primero veamos un formulario que envía parámetros a otro formulario utilizando el método POST:

El formulario que envía se denomina en este caso “envia.php” y tiene el siguiente código:

```
<HTML>

<HEAD>
<TITLE>USANDO EL MÉTODO POST
</TITLE>
</HEAD>

<BODY>
```

```
<FORM name="formulario1" ACTION="recibe.php" METHOD="POST">
<TABLE>
  <TR>
    <TD COLSPAN=2>
      <INPUT TYPE="text" NAME="BOX1" VALUE="CAJA1">
    </TD>
  </TR>
  <TR>
    <TD COLSPAN=2>
      <INPUT TYPE="text" NAME="BOX2" VALUE="CAJA2">
    </TD>
  </TR>
  <TR>
    <TD COLSPAN=2>
      <INPUT TYPE="text" NAME="BOX3" VALUE="CAJA3">
    </TD>
  </TR>
  <TR>
    <TD COLSPAN=2>
      <INPUT TYPE="text" NAME="BOX4" VALUE="CAJA4">
    </TD>
  </TR>

  <TR>
    <TD>
      <INPUT TYPE="reset" VALUE="CANCELAR"">
    </TD>
    <TD>
      <INPUT TYPE="submit" VALUE="ACEPTAR"">
    </TD>
  </TR>

</TABLE>
</FORM>

</BODY>
</HTML>
```

En términos de interfaz de usuario, la página anterior se puede ver muy similar a la siguiente:

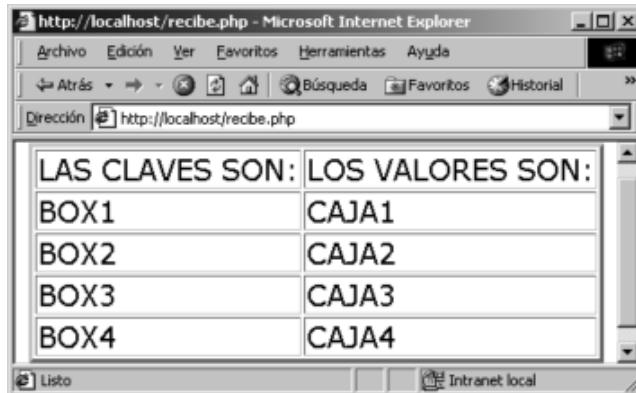


Cuando el usuario hace clic sobre el botón “Aceptar” de la figura anterior, se llama a la página denominada “recibe.php”, que tiene el siguiente código:

```
<HTML>
<BODY>
<TABLE BORDER = 3>
<TR>
<TD>
LAS CLAVES SON:
</TD>
<TD>
LOS VALORES SON:
</TD>
</TR>
<?php
foreach ($HTTP_POST_VARS as $clave => $valor){
echo "<TR>";
echo "<TD>";
echo "$clave \n";
echo "</TD>";
echo "<TD>";
echo "$valor \n";
echo "</TD>";
echo "</TR>";
```

```
}  
?>  
</TABLE>  
</BODY>  
<HTML>
```

Al ser llamada desde la página “envia.php”, muestra la siguiente página al usuario navegante:



Es importante que el lector observe que el navegador muestra en la barra de direcciones el siguiente enlace:

<http://localhost/recibe.php>

lo que quiere decir que al navegante de la aplicación le es imposible ver los nombres y los valores de las variables que fueron pasadas a la página denominada “recibe.php”. Más adelante, como el lector podrá observar, se mostrará que cuando se pasan parámetros con el método POST de un formulario, la dirección mostrada por el navegador web expresa unos parámetros y unos valores que en realidad coinciden con los valores y variables que se están pasando a otra página.

Pasar parámetros en formularios con el método GET

A continuación se muestra casi el mismo ejercicio anterior, que se hizo utilizando paso de parámetros en formulario con el método POST, pero ahora con el método GET. En otras palabras, se va a cambiar una pequeña línea de código al ejemplo anterior, con el fin de ver la diferencia radical entre utilizar el método POST y el método GET.

Para comenzar veamos el código de la página que tiene los datos, y que para este caso se denomina la página “envia1.php”:

```

<HTML>

<HEAD>
<TITLE>USANDO EL MÉTODO POST
</TITLE>
</HEAD>

<BODY>
<FORM name="formulario1" ACTION="recibe1.php" METHOD="GET">
<TABLE>
<TR>
<TD COLSPAN=2>
<INPUT TYPE="text" NAME="BOX1" VALUE="CAJA1">
</TD>
</TR>
<TR>
<TD COLSPAN=2>
<INPUT TYPE="text" NAME="BOX2" VALUE="CAJA2">
</TD>
</TR>
<TR>
<TD COLSPAN=2>
<INPUT TYPE="text" NAME="BOX3" VALUE="CAJA3">
</TD>
</TR>
<TR>
<TD COLSPAN=2>
<INPUT TYPE="text" NAME="BOX4" VALUE="CAJA4">
</TD>
</TR>

<TR>
<TD>
<INPUT TYPE="reset" VALUE="CANCELAR"">
</TD>
<TD>
<INPUT TYPE="submit" VALUE="ACEPTAR"">

```

```
</TD>
</TR>

</TABLE>
</FORM>

</BODY>
</HTML>
```

Esta página se verá con el siguiente aspecto:



Cuando el usuario hace clic sobre el botón “Aceptar”, se llama a la página denominada “recibe1.php”, la cual tiene el siguiente código:

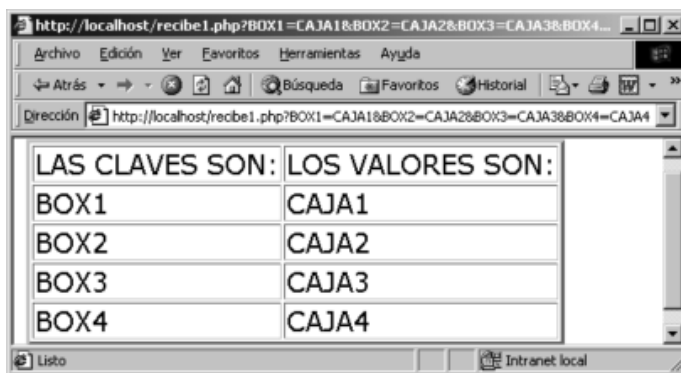
```
<HTML>
<BODY>
<TABLE BORDER = 3>
<TR>
<TD>
LAS CLAVES SON:
</TD>
<TD>
LOS VALORES SON:
</TD>
</TR>
<?php
foreach ($HTTP_GET_VARS as $clave => $valor){
```

```

echo "<TR>";
echo "<TD>";
echo "$clave \n";
echo "</TD>";
echo "<TD>";
echo "$valor \n";
echo "</TD>";
echo "</TR>";
}
?>
</TABLE>
</BODY>
<HTML>

```

Al ser llamada desde la página “envia1.php”, se ve en el navegador de la siguiente manera:



Aquí el lector puede observar que la barra de dirección del navegador contiene el siguiente enlace:

```
http://localhost/recibe1.php?BOX1=CAJA1&BOX2=CAJA2&BOX3=CAJA3&BOX4=CAJA4
```

Pasar parámetro con botones e hiperenlaces

Es muy importante que el lector comprenda cómo se pasan parámetros, es decir, cómo se pasan valores de variables de una página a otra. En el siguiente ejercicio se indica cómo se pueden utilizar algunas funciones de JavaScript con el ánimo de obtener los contenidos de las cajas de texto y pasarlos como parámetros a cada uno de los botones. La otra ventaja del siguiente ejercicio es que se sale de lo mostrado

habitualmente por los libros, que casi siempre indican cómo se maneja un botón con la función “submit” y de pronto otro con la función “reset”, pero muy pocas veces explican cómo se pueden utilizar más botones en los formularios.

Veamos el código:

```
<HTML>

<HEAD>
<TITLE>MANEJANDO VARIOS BOTONES</TITLE>
<SCRIPT language=JavaScript>
<!--
function uno() {
top.location.href = 'primero.php?V1='+document.formulario1.BOX1.value;
}

function dos() {
top.location.href = 'segundo.php?V1='+document.formulario1.BOX2.value;
}

function tres() {
top.location.href = 'tercero.php?V1='+document.formulario1.BOX3.value;
}

function cuatro() {
top.location.href = 'cuarto.php?V1='+document.formulario1.BOX4.value;
}
-->
</SCRIPT>
</HEAD>

<BODY>
<FORM name="formulario1">
<TABLE>
<TR>
<TD>
<INPUT TYPE="text" NAME="BOX1" VALUE="CAJA1">
</TD>
<TD>
```



```

<INPUT TYPE="text" NAME="BOX2" VALUE="CAJA2">
</TD>
<TD>
<INPUT TYPE="text" NAME="BOX3" VALUE="CAJA3">
</TD>
<TD>
<INPUT TYPE="text" NAME="BOX4" VALUE="CAJA4">
</TD>
</TR>

<TR>
<TD>
<INPUT TYPE="button" VALUE="PRIMIS" onClick="uno()">
</TD>
<TD>
<INPUT TYPE="button" VALUE="SEGUNDIS" onClick="dos()">
</TD>
<TD>
<INPUT TYPE="button" VALUE="TERCERIS" onClick="tres()">
</TD>
<TD>
<INPUT TYPE="button" VALUE="CUARTIS" onClick="cuatro()">
</TD>
</TR>

<TR>
<TD>
<A HREF="primero.php?V1=LUIS">REFERENCIA UNO</A>
</TD>
<TD>
<A HREF="segundo.php?V1=FELIPE">REFERENCIA DOS</A>
</TD>
<TD>
<A HREF="tercero.php?V1=WANUMEN">REFERENCIA TRES</A>
</TD>
<TD>

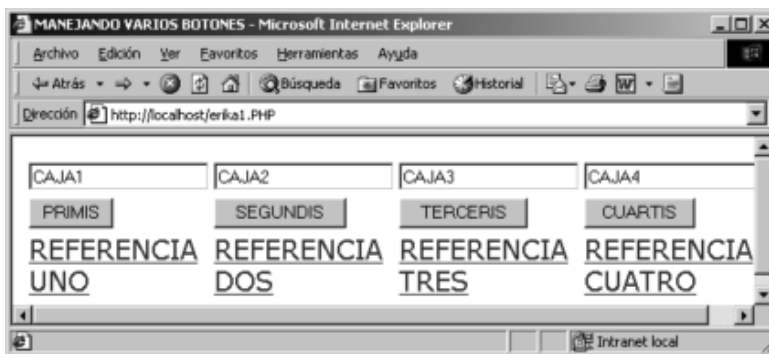
```

```
<A HREF="cuarto.php?V1=SILVA">REFERENCIA CUATRO</A>
</TD>
</TR>
</TABLE>
</FORM>

</BODY>

</HTML>
```

El anterior código produce una página similar a la siguiente:



(El nombre de esta página se debe a que fue una estudiante llamada Erika la que me propuso el ejercicio y me dió la tarea de resolverlo).

Como el lector puede apreciar, se tiene una página principal llamada “erika1.php”, que contiene cuatro botones que permiten desplazarse a una página determinada. Por ejemplo, al hacer clic sobre el botón “Primis” es posible desplazarnos a la página denominada “primero.php”, al hacer clic sobre el botón con el rótulo “Segundis” es posible desplazarnos a la página denominada “segundo.php”, y así sucesivamente.

Ahora vamos a examinar el código de la página “primero.php”:

```
<?PHP
echo 'AHORA ESTOY AQUI' .$V1;
?>
```

En realidad el código de la página “segundo.php” es

```
<?PHP
echo 'AHORA ESTOY AQUI' .$V1;
?>
```

El de la página “tercero.php” es

```
<?PHP
echo 'AHORA ESTOY AQUÍ' . $V1;
?>
```

y el de la página “cuatro.php” es

```
<?PHP
echo 'AHORA ESTOY AQUÍ' . $V1;
?>
```

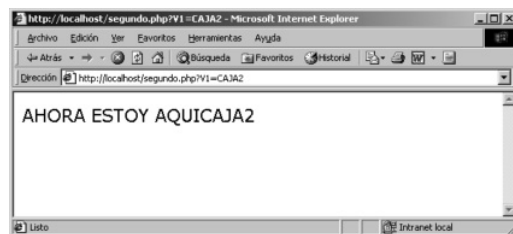
Con lo anterior, el lector puede apreciar que no hay diferencia en el código de dichas páginas. Lo que nos interesa mostrar es que cada una de las páginas mencionadas imprimen el valor de la variable “V1” y en realidad dicha variable es pasada como parámetro en la función JavaScript respectiva que maneja el evento clic sobre dicho botón.

Por ejemplo, si pulsamos el primer botón se ejecuta la función:

```
function uno() {
top.location.href = 'primero.php?V1='+document.formulario1.BOX1.value;
}
```

Es decir que el navegador se desplaza hasta la página denominada “primero.php” y le pasa un valor a la variable “V1”. Dicho valor es realmente el valor digitado por el usuario en la caja de texto denominada “BOX1”.

Ahora bien, con el ánimo de lograr una mayor comprensión de este ejercicio por parte del lector, vamos a suponer que el usuario hace clic sobre el botón cuyo rótulo es “Segundis”, con lo cual el navegador, además de ir a la página denominada “segundo.php”, nos mostrará un resultado similar al siguiente.



Creo que no es necesario hacer la misma explicación para cada uno de los botones, puesto que funcionan de manera análoga al botón que acabamos de explicar, a diferencia de los enlaces que funcionan un poco diferente más en sintaxis que en esencia.

Para no ir más lejos con explicaciones, supongamos que el usuario hace clic sobre el tercer hipervínculo, es decir “Referencia tres”, con lo cual el navegador, además de ir a la página denominada “tercero.php”, muestra un resultado similar al siguiente.



Para lograr una mayor comprensión de este ejercicio, detallemos que el parámetro dentro de la función JavaScript se pasa de la siguiente manera:

```
top.location.href = 'primero.php?V1='+document.formulario1.BOX1.value;
```

mientras que en un hipervínculo se pasa de la siguiente forma:

```
<A HREF="primero.php?V1=LUIS">REFERENCIA UNO</A>
```

Es importante notar que la diferencia radica principalmente en la ubicación de las comillas dobles y sencillas, cosa que de no hacerse tal como se acaba de mostrar, no funcionaría correctamente.

Pasar varios parámetro con botones

En el anterior ejercicio se mostró cómo se puede pasar un parámetro cuando el usuario hace clic sobre un botón o sobre un hipervínculo. Pues bien, ahora se explicará cómo es posible pasar varios parámetros cuando el usuario hace clic sobre un solo botón. Para esto vamos a suponer un formulario muy sencillo, en el cual el usuario llena los nombres de una persona.

Dicho formulario tiene un aspecto similar al siguiente:

A screenshot of a web browser window titled 'MANEJANDO VARIOS BOTONES - Microsoft Internet Explorer'. The address bar shows 'http://localhost/ERIKA2.PHP'. The form contains four text input fields with labels: 'PRIMER NOMBRE:', 'SEGUNDO NOMBRE:', 'PRIMER APELLIDO:', and 'SEGUNDO APELLIDO:'. Each field has a placeholder text 'CA.JA1', 'CA.JA2', 'CA.JA3', and 'CA.JA4' respectively. Below the input fields are four buttons: 'NOMBRES', 'APELLIDOS', 'NOMBRE CORTO', and 'COMPLETO'. The status bar at the bottom shows 'Listo' and 'Intranet local'.

Y fue elaborado con el siguiente código:

```
<HTML>

<HEAD>
<TITLE>MANEJANDO VARIOS BOTONES</TITLE>
<SCRIPT language=JavaScript>
<!--
function uno() {
top.location.href

='A.PHP?V1='+document.f1.BOX1.value+'&V2='+document.f1.BOX2.value;
}

function dos() {
top.location.href

='B.PHP?V1='+document.f1.BOX3.value+'&V2='+document.f1.BOX4.value;
}

function tres() {
top.location.href

='C.PHP?V1='+document.f1.BOX1.value+'&V2='+document.f1.BOX3.value;
}

function cuatro() {
top.location.href

='D.PHP?V1='+document.f1.BOX1.value+'&V2='+document.f1.BOX2.va-
lue+'&

V3='+document.f1.BOX3.value+'&V4='+document.f1.BOX4.value;
}
-->
</SCRIPT>
</HEAD>
<BODY>
```

```
<FORM name="f1">
<TABLE>
<TR>
<TD>
PRIMER NOMBRE:
</TD>
<TD>
<INPUT TYPE="text" NAME="BOX1" VALUE="CAJA1">
</TD>
</TR>
<TR>
<TD>
SEGUNDO NOMBRE:
</TD>
<TD>
<INPUT TYPE="text" NAME="BOX2" VALUE="CAJA2">
</TD>
</TR>
<TR>
<TD>
PRIMER APELLIDO:
</TD>
<TD>
<INPUT TYPE="text" NAME="BOX3" VALUE="CAJA3">
</TD>
</TR>
<TR>
<TD>
SEGUNDO APELLIDO:
</TD>
<TD>
<INPUT TYPE="text" NAME="BOX4" VALUE="CAJA4">
</TD>
</TR>

<TR>
<TD COLSPAN=4>
```

```

<CENTER>
<INPUT TYPE="button" VALUE="NOMBRES" onClick="uno()">
<INPUT TYPE="button" VALUE="APELLIDOS" onClick="dos()">
<INPUT TYPE="button" VALUE="NOMBRE CORTO" onClick="tres()">
<INPUT TYPE="button" VALUE="COMPLETO" onClick="cuatro()">
</CENTER>
</TD>
</TR>

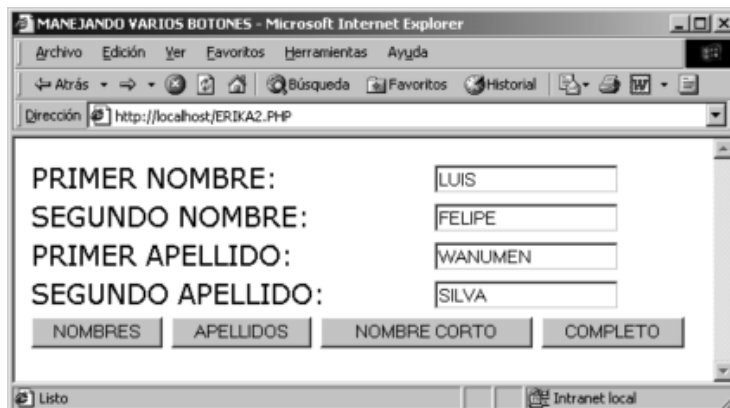
</TABLE>
</FORM>

</BODY>

</HTML>

```

Suponemos que el usuario borra los valores que aparecen por defecto en las cajas de texto y coloca los nombres y apellidos de una persona. Por ejemplo, presumamos que el usuario coloca el nombre “Luis Felipe Wanumen Silva”, con lo cual nuestro formulario tendrá una apariencia similar a la siguiente.



Ahora vamos a suponer que el usuario hace clic sobre el botón “Nombres”. Observamos que el navegador intenta desplazarse a la página “A.PHP” y la pasa dos parámetros: V1 y V2, que en realidad equivalen a lo que hay en las cajas BOX1 y BOX2 respectivamente.

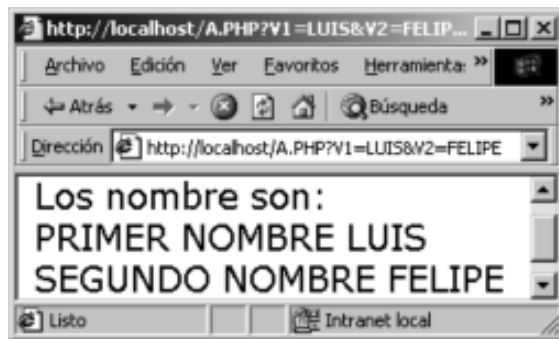
En la barra de direcciones del navegador se puede apreciar esta situación, pues el navegador llama a la página “A.PHP” de la siguiente manera:

```
http://localhost/A.PHP?V1=LUIS&V2=FELIPE
```

y para lograr una mejor comprensión de este ejercicio por parte del lector, veamos el código de la página “A.PHP”:

```
<HTML>
<BODY>
<?PHP
echo 'Los nombre son: ' . '<BR>';
echo 'PRIMER NOMBRE ' . $V1 . '<BR>';
echo 'SEGUNDO NOMBRE ' . $V2;
?>
</BODY>
</HTML>
```

Con lo anterior podremos darnos cuenta que al hacer clic sobre el primer botón del formulario que estamos trabajando, la página “A.PHP” tomará un aspecto similar al siguiente:



Ahora veamos el código de las páginas “B.PHP”, “C.PHP” y “D.PHP”.

Código de la página “B.PHP”:

```
<HTML>
<BODY>
<?PHP
echo 'Los apellidos son: ' . '<BR>';
echo 'PRIMER APELLIDO ' . $V1 . '<BR>';
echo 'SEGUNDO APELLIDO ' . $V2;
?>
</BODY>
</HTML>
```

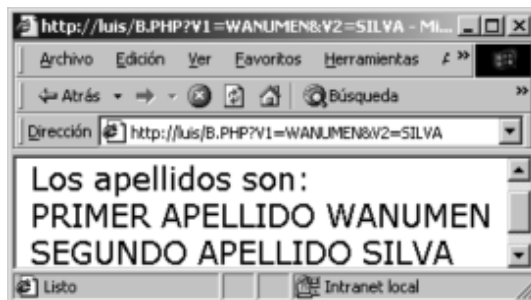

Código de la página “C.PHP”:

```
<HTML>
<BODY>
<?PHP
echo 'El nombre corto es : '.'<BR>';
echo 'PRIMER NOMBRE ' . $V1 . '<BR>';
echo 'PRIMER APELLIDO ' . $V2;
?>
</BODY>
</HTML>
```

Código de la página “D.PHP”:

```
<HTML>
<BODY>
<?PHP
echo 'El nombre completo es : '.'<BR>';
echo 'PRIMER NOMBRE ' . $V1 . '<BR>';
echo 'SEGUNDO NOMBRE ' . $V2 . '<BR>';
echo 'PRIMER APELLIDO ' . $V3 . '<BR>';
echo 'SEGUNDO APELLIDO ' . $V4;
?>
</BODY>
</HTML>
```

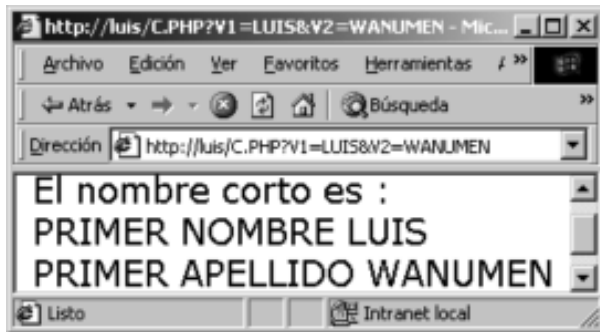
Para finalizar con nuestra explicación, veamos las páginas mostradas en el navegador cuando el usuario presiona el botón “Apellidos”:



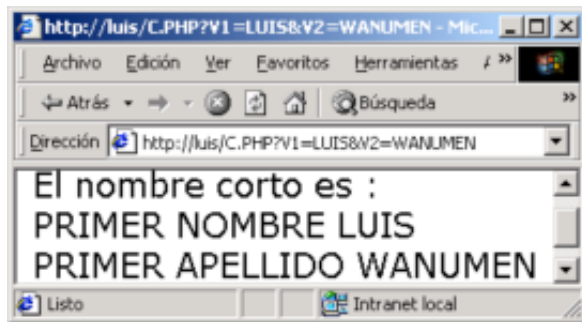
En este caso, el nombre del servidor es “luis” y en las páginas anteriores el nombre del servidor ha sido “localhost”. La explicación es muy sencilla, puesto que “local-

host” es un nombre estándar para cualquier servidor, que hace referencia al servidor actual en el que se está trabajando.

Cuando el usuario presiona el botón con la etiqueta “Nombre corto” vemos:



Y finalmente, cuando el usuario presiona el botón con la etiqueta “Completo” vemos:



Espero que con este ejercicio el lector haya comprendido la esencia del manejo de aplicaciones web que pasan parámetros de una página a otra utilizando PHP como lenguaje en el servidor.

Insertar datos pidiendo confirmación

Para ejecutar el siguiente ejercicio es bueno crear la base de datos denominada “luchin”, la cual puede ser creada con la instrucción:

```
create database luchin
```

Después de esto, es necesario que el lector cree una tabla denominada “alumno”, la cual puede ser creada con el *script* mostrado a continuación:

```
CREATE TABLE `alumno` (  
  `cedula` INT NOT NULL ,  
  `nombre` VARCHAR( 30 ) NOT NULL ,
```

```
`apellido` VARCHAR( 30 ) NOT NULL ,
`edad` INT NOT NULL ,
`sexo` VARCHAR( 30 ) NOT NULL );
```

Después se puede crear un usuario denominado “felipito”, al que se le conceden privilegios sobre la base de datos “luchin”.

Ahora el lector puede crear una página PHP llamada “abrir1.php”, la cual tiene el siguiente código:

```
<HTML>
<HEAD>

<script language="JavaScript">


function cerrar() {
top.location.href = 'javascript:window.close()';
}


function arbirVentana () {
var cedulilla = document.form1.cedula1.value;
var nombrecillo = document.form1.nombre1.value;
var apellidillo = document.form1.apellido1.value;
var edadcilla = document.form1.edad1.value;
var sexillo = document.form1.sexo1.value;

msg=open("", "DisplayWindow", "toolbar=no,directories=no,menubar=no");
msg.document.write("<html>");
msg.document.write("<HEAD><TITLE>MENSAJE DE CONFIRMA-
CION</TITLE>");

// <a href='javascript:window.close()'>pulsando aquí</a>
msg.document.write("      <script language='JavaScript' ">);
msg.document.write(">");
msg.document.write("      function noEnviar () { ">);
msg.document.write(" window.close() ">);
msg.document.write("    } ">);
msg.document.write("      function siEnviar () { ">);
```

```
cedulis = cedulilla;
nombris = nombrecillo;
apellidis = apellidillo;
edadis = edadcilla;
sexillis = sexillo;

msg.document.write("top.location.href = 'abrir2.php?cedula= ";
msg.document.write(cedulis);
msg.document.write("&nombre=");
msg.document.write(nombris);
msg.document.write("&apellido=");
msg.document.write(apellidis);
msg.document.write("&edad=");
msg.document.write(edadis);
msg.document.write("&sexo=");
msg.document.write(sexillis);

msg.document.write(" '");
msg.document.write("    } ");

msg.document.write("    </script");
msg.document.write("> ");

msg.document.write("</HEAD>");
msg.document.write("<body>");
msg.document.write("<table>");

msg.document.write(" <tr>");
msg.document.write("« <td colspan = '2'>»");
msg.document.write("« Desea enviar sus datos?»");
msg.document.write("« </td>»");
msg.document.write("« </tr>»");

msg.document.write("« <tr>»");
msg.document.write("« <td>»");
```

```

msg.document.write(« <input type = 'reset' name = 'Boton1' value = 'NO' on
Click= 'noEnviar()'>»);
msg.document.write(“ </td>”);
msg.document.write(“ <td>”);
msg.document.write(“ <input type='button' name='Boton2' value='SI' onclick=
'siEnviar ()'>”);
msg.document.write(“ </td>”);
msg.document.write(“ </tr>”);

msg.document.write(“</table>”);
msg.document.write(“</body>”);
msg.document.write(“</html>”);

}
</script>
</HEAD>
<BODY>

<center>
<table border = 5>
<tr>
<td colspan = 2>Llena tu hoja de vida:
</td>
</tr>
<form name = “form1”>
<tr>
<td>Cedula:
</td>
<td><input type = “text” name = “cedula1”>
</td>
</tr>
<tr>
<td>Nombre:
</td>
<td><input type = «text» name = «nombre1»>
</td>

```

```
</tr>
<tr>
<td>Apellido:
</td>
<td><input type = "text" name = "apellido1">
</td>
</tr>
<tr>
<td>Edad:
</td>
<td><input type = "text" name = "edad1">
</td>
</tr>
<tr>
<td>Sexo:
</td>
<td><input type = "text" name = "sexo1">
</td>
</tr>
<tr>
<td><input type = "reset" name = "Boton1" value = "Cancelar">
</td>
<td><input type="button" name="Boton2" value="Insertar" onclick=" arbir-
Ventana ()">
</td>
</tr>
<form>
</table>
</center>

</BODY>
</HTML>
```

La anterior página, en términos de interfaz de usuario, toma la siguiente apariencia:



http://localhost/abrir...

Archivo Edición Ver Favorit

Atrás Vínculos

Llena tu hoja de vida:

Cedula: 79904602

Nombre: Luis

Apellido: Silva

Edad: 28

Sexo: Masculino

Cancelar Insertar

Intranet local

Cuando el usuario hace clic sobre el botón con el rótulo “Cancelar”, se obtiene el siguiente resultado:



http://localhost/abrir...

Archivo Edición Ver Favorit

Atrás Vínculos

Llena tu hoja de vida:

Cedula:

Nombre:

Apellido:

Edad:

Sexo:

Cancelar Insertar

Intranet local

Es decir, se blanquean las cajas de texto, lo que básicamente se debe a que dicho botón es de tipo “reset”. Ahora bien, supongamos que el usuario llena los datos, por ejemplo los datos del autor del manual:



http://localhost/abrir...

Archivo Edición Ver Favorit

Atrás Vínculos

Llena tu hoja de vida:

Cedula:	87654321
Nombre:	Luis
Apellido:	Silva
Edad:	28
Sexo:	Masculino

Cancelar Insertar

Intranet local

y presiona el botón con el rótulo “Insertar”. Se tiene entonces el siguiente resultado:



MENSAJE DE CONFIRMACION - Micros...

Desea enviar sus datos?

NO SI

Se abre una página de confirmación, la cual, valga la redundancia, pregunta nuevamente si el usuario desea insertar los datos.

Si el usuario hace clic en el botón “No”, se cierra la ventana de confirmación y se devuelve el control a la ventana inicial donde se habían insertado los datos. Por otra parte, si el usuario hace clic en el botón “Sí”, el servidor llama a la página “abrir2.php”, la cual tiene el siguiente código:

```
<html>
<body>
<table>
<tr>
<td colspan = 2>Al insertar los datos:
</td>
</tr>
```



```

<tr>
<td>Cedula:
</td>
<td><?= $cedula ?>
</td>
</tr>

<tr>
<td>Nombre:
</td>
<td><?= $nombre ?>
</td>
</tr>

<tr>
<td>Apellido:
</td>
<td><?= $apellido ?>
</td>
</tr>

<tr>
<td>Edad:
</td>
<td><?= $edad ?>
</td>
</tr>

<tr>
<td>Sexo:
</td>
<td><?= $sexo ?>
</td>
</tr>

<tr>
<td colspan = 2>Se presentaron los detalles:
</td>

```

```

</tr>

<tr>
<td colspan = 2>

<?php
$errores = 0;
if(!($conexion=mysql_connect("luisito","felipito","felipito")))
{
echo "Errores conectando con el servidor";
$errores = 1;
exit();
}
else
{
echo "Conexion correcta". "<br>";
}

if (!mysql_select_db("luchin",$conexion))
{
echo "Error conectando con la base de datos.";
$errores = 1;
echo "<br>";
exit();
}
else{
echo "Base de datos correcta";
echo "<br>";
}

$cadena = "insert into alumno values(";
$cadena = $cadena . $cedula . ",";
$cadena = $cadena . " " . $nombre . " ";
$cadena = $cadena . " " . $apellido . " ";
$cadena = $cadena . $edad . ",";
$cadena = $cadena . " " . $sexo . " ";

$result=mysql_query($cadena,$conexion);
mysql_free_result($result);

```

```

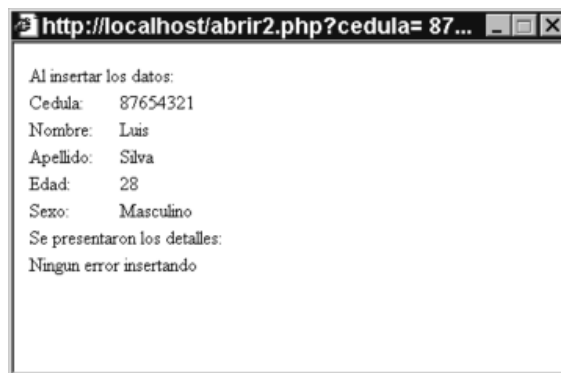
mysql_close($conexion);
echo "Ningun error insertando";
?>

</td>
</tr>

</table>
</body>
</html>

```

Esta página, al ser procesada desde la página anterior con los datos que estamos trabajando, produce el siguiente resultado:

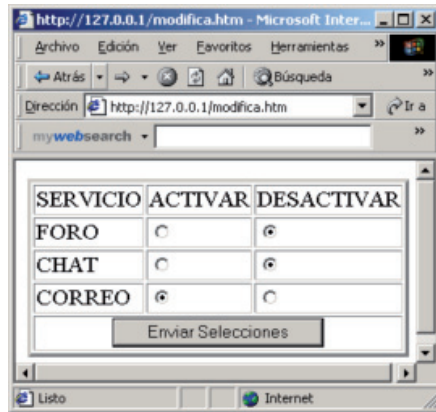


Este es un ejercicio clásico de programación, el cual pudo haber sido desarrollado con otras técnicas, como la de los cuadros de diálogo en JavaScript, sin embargo la técnica usada se ejecuta mejor en la gran mayoría de los navegadores por ser independiente de la configuración de JavaScript que tenga cada navegador.

Pasar opciones en un formulario

En esta sección se pretende mostrar cómo por medio de una página HTML es posible crear opciones que puedan ser escogidas por los usuarios, de tal forma que la página PHP que reciba los datos pueda saber a ciencia cierta qué opciones fueron escogidas. Para este caso en particular se va a mostrar la posibilidad de que el usuario pueda activar o desactivar unos servicios, de tal suerte que la página PHP que reciba los datos pueda determinar cuáles opciones están activadas y cuáles fueron desactivadas.

Para comprender mejor esta situación empecemos por mostrar en forma gráfica la interfaz que se quiere enseñar al cliente:



En este caso el usuario escogió desactivar las opciones de foro y de chat, en tanto que eligió activar la opción de correo. Para observar cómo se hace esta página, veamos el código fuente:

```
<HTML>
<HEAD>
</HEAD>
<BODY>
<SCRIPT language=JavaScript type=text/javascript>
<!--
function Valida_forma(forma)
{

    var seleccionado = false;
    for (i = 0; i < forma.Pregunta_1.length; i++)
    {
        if (forma.Pregunta_1[i].checked)
            seleccionado = true;
    }
    if (!seleccionado)
    {
        alert("Seleccione una opción de la Pregunta 1");
        return (false);
    }

    var seleccionado = false;
```

```

for (i = 0; i < forma.Pregunta_2.length; i++)
{
if (forma.Pregunta_2[i].checked)
seleccionado = true;
}
if (!seleccionado)
{
alert("Seleccione una opción de la Pregunta 2");
return (false);
}

return (true);
}
//--></SCRIPT>

<FORM action=modifica1.php language=JavaScript
method=get name=EVALUACION onsubmit="return Valida_forma(this)">
<TABLE border=3>
<TBODY>
<TR>
<TD>SERVICIO </TD>
<TD>ACTIVAR </TD>
<TD>DESACTIVAR </TD></TR>
<TR>
<TD>FORO </TD>
<TD><INPUT name=Pregunta_1 type=radio value=0> </TD>
<TD><INPUT name=Pregunta_1 type=radio value=1> </TD></TR>
<TR>
<TD>CHAT </TD>
<TD><INPUT name=Pregunta_2 type=radio value=0> </TD>
<TD><INPUT name=Pregunta_2 type=radio value=1> </TD></TR>
<TR>
<TD>CORREO </TD>
<TD><INPUT name=Pregunta_3 type=radio value=0> </TD>
<TD><INPUT name=Pregunta_3 type=radio value=1> </TD></TR>
<TR>

```

```
<TD colSpan=3>
<CENTER><INPUT name=B1 type=submit value="Enviar Selecciones">
</CENTER>
</TD>
</TR>
</TBODY>
</TABLE>
</FORM>
</BODY>
</HTML>
```

Esta página verifica que el usuario haya elegido por lo menos una de las opciones. Si no ha elegido una opción, inmediatamente el JavaScript que se encuentra en la página como producto de procesar la función “Valida_forma” retorna falso y por lo tanto no deja enviar los datos a la página especificada en “action”. En este caso no enviaría los datos a la página “modifica1.php”.

La idea es que la página PHP que recibe los datos sea una página que conozca qué servicios son solicitados para activar y qué servicios son solicitados para desactivar. A manera de ejemplo, basándonos en las opciones elegidas anteriormente por el usuario, tendríamos que la página PHP que recibe los datos mostraría un mensaje similar al siguiente:



Lo cual se lograría con el siguiente código:

```
<html>
</body>
<?php
    $var1 =$Pregunta_1;
```

```
if($Pregunta_1==0){
echo "<br>Foro Activado<br>";
}
else{
echo "<br>Foro Desactivado<br>";
}

$var2 =$Pregunta_2;
if($Pregunta_2==0){
echo "<br>Chat Activado<br>";
}
else{
echo "<br>Chat Desactivado<br>";
}

$var3 =$Pregunta_3;
if($Pregunta_3==0){
echo "<br>Correo Activado<br>";
}
else{
echo "<br>Correo Desactivado<br>";
}
?>
</body>
</html>
```


Ejemplo de una búsqueda _____

Objetivo general

El objetivo general de este ejemplo es hacer un primer ejercicio en PHP que permita buscar un registro almacenado previamente en la base de datos MySQL.

Objetivo específico

- Crear las tablas en la base de datos para el ejercicio de búsqueda.
- Insertar los registros necesarios en las tablas.
- Realizar ocho ejercicios de consultas para manejar consultas de selección.
- Desarrollar la aplicación PHP que permita la búsqueda.

Metodología

Básicamente, de 8 a 10 a. m. se expone en el tablero la secuencia de los temas que se van a realizar en el curso. De 10 a. m. a 12 m. se exponen los fundamentos del lenguaje HTML en el tablero. De 1 a 3 p. m. se hace un ejercicio estilo conferencia para que los estudiantes lo puedan ver con Netmeeting y se resuelven las dudas sobre dicho ejercicio. De 3 p. m. a 5 p. m. se deja un ejercicio tipo taller para que los estudiantes lo desarrollen.

Desarrollo de la clase

La clase se desarrolla con los siguientes contenidos:

- Creación de las tablas en la base de datos
- Inserción de registros en las tablas
- Ejercicios de consultas
- Primera consulta
- Segunda consulta

- Tercera consulta
- Cuarta consulta
- Quinta consulta
- Sexta consulta
- Séptima consulta
- Octava consulta
- Aplicación PHP

Recursos

Para la parte inicial, correspondiente a la explicación, se requiere un tablero. Para la parte práctica se requiere como mínimo una sala de informática donde el número de estudiantes por computador puede ser máximo dos.

Bibliografía

- Delisle, M. (2007). *Dominar phpMyAdmin para una administración efectiva de MySQL*. Quebec-Canadá Packt Publishing.
- Heurtel, O. (2009). *PHP y MySQL. Domine el desarrollo de un sitio web dinámico e interactivo*. Barcelona: Ediciones ENI.
- Suehring, S.; Converse, T. y Park, J. (2009). *PHP 6 and MySQL Bible*. Indiana: Wiley Pub.

Creación de las tablas en la base de datos

```
create table estud20(  
  codest int primary key,  
  nomest varchar(50)  
);  
  
create table mat20(  
  codmat int primary key,  
  nommat varchar(50)  
);  
  
create table not20(  
  codnot int primary key,  
  codest int,
```

```
codmat int,
valor int
);
```

Inserción de registros en las tablas

```
insert into estud20 values(1,'Luis');
insert into estud20 values(2,'Juan');
insert into estud20 values(3,'Luis Felipe');
insert into estud20 values(4,'Carlos');
insert into estud20 values(5,'Mario');
```

```
insert into mat20 values(1,'Matematicas');
insert into mat20 values(2,'Fisica');
insert into mat20 values(3,'Religion');
insert into mat20 values(4,'Historia');
```

```
insert into not20 values(1,1,1,4);
insert into not20 values(2,1,2,3);
insert into not20 values(3,1,3,5);
insert into not20 values(4,2,2,5);
```

Ejercicios de consultas

Basado en el esquema de tablas anterior, se propone la generación de algunas consultas que pueden realizarse y probarse gracias a la información previamente ingresada en la base de datos. En esta sección se explican al detalle ocho posibles consultas que se pueden realizar con el esquema de bases de datos elaborado en MySQL.

Primer ejercicio

Consulta:

```
Select * from estud20;
```

Resultado:

1	Luis
2	Juan
3	Luis Felipe
4	Carlos
5	Mario

Segundo ejercicio

Consulta:

```
select * from estud20 where codest = 1;
```

Resultado:

1	Luis
---	------

Tercer Ejercicio

Consulta:

```
Select * from not20, estud20;
```

Resultado:

codnot	codest	codmat	valor	codest	nomest
1	1	1	4	1	Luis
2	1	2	3	1	Luis
3	1	3	5	1	Luis
4	2	2	5	1	Luis
1	1	1	4	2	Juan
2	1	2	3	2	Juan
3	1	3	5	2	Juan
4	2	2	5	2	Juan
1	1	1	4	3	Luis Felipe
2	1	2	3	3	Luis Felipe
3	1	3	5	3	Luis Felipe
4	2	2	5	3	Luis Felipe
1	1	1	4	4	Carlos
2	1	2	3	4	Carlos
3	1	3	5	4	Carlos
4	2	2	5	4	Carlos
1	1	1	4	5	Mario
2	1	2	3	5	Mario
3	1	3	5	5	Mario
4	2	2	5	5	Mario

El lector debe observar que existen algunos valores que no son válidos, y por lo tanto se hace necesario ver únicamente los que tengan unos valores iguales tanto en el campo “codest” de la tabla estudiante, como en el campo “codest” de la tabla notas.

La siguiente consulta mejora el ejercicio.

Cuarto ejercicio

Consulta:

```
Select *
from not20, estud20
where not20.codest = estud20.codest;
```

Resultado:

codnot	codest	codmat	valor	codest	nomest
1	1	1	4	1	Luis
2	1	2	3	1	Luis
3	1	3	5	1	Luis
4	2	2	5	2	Juan

Quinto ejercicio

De los campos de la anterior consulta, solo mostrar los que estan en negrilla:

codnot	codest	codmat	valor	codest	nomest
1	1	1	4	1	Luis
2	1	2	3	1	Luis
3	1	3	5	1	Luis
4	2	2	5	2	Juan
NOT20				ESTUD20	

Esto se logra con la siguiente consulta:

```
Select not20.codest, not20.codmat, not20.valor,estud20.nomest
from not20, estud20
where not20.codest = estud20.codest;
```

Resultado:

codest	codmat	valor	nomest
1	1	4	Luis
1	2	3	Luis
1	3	5	Luis
2	2	5	Juan

Sexto ejercicio

Incorporamos la tabla materia:

```
Select not20.codest, not20.codmat, not20.valor,estud20.nomest, mat20.codmat,
mat20.nommat
from not20, estud20, mat20
where not20.codest = estud20.codest;
```

Resultado:

codest	codmat	valor	nomest	codmat	nommat
1	1	4	Luis	1	Matemáticas
1	2	3	Luis	1	Matemáticas
1	3	5	Luis	1	Matemáticas
2	2	5	Juan	1	Matemáticas
1	1	4	Luis	2	Física
1	2	3	Luis	2	Física
1	3	5	Luis	2	Física
2	2	5	Juan	2	Física
1	1	4	Luis	3	Religión
1	2	3	Luis	3	Religión
1	3	5	Luis	3	Religión
2	2	5	Juan	3	Religión
1	1	4	Luis	4	Historia
1	2	3	Luis	4	Historia
1	3	5	Luis	4	Historia
2	2	5	Juan	4	Historia

Observe que sale un poco de basura, que es necesario depurar como lo muestra la siguiente consulta.

Septimo ejercicio

Consulta:

```
Select not20.codest, not20.codmat, not20.valor,estud20.nomest, mat20.codmat,
mat20.nommat
from not20, estud20, mat20
where not20.codest = estud20.codest
and not20.codmat = mat20.codmat
```

Resultado:

codest	codmat	valor	nomest	codmat	nommat
1	1	4	Luis	1	Matematicas
1	2	3	Luis	2	Fisica
2	2	5	Juan	2	Fisica
1	3	5	Luis	3	Religion

Para mejorar la consulta mostramos una sola vez el código de materia. Para esto realizamos la siguiente consulta.

Octavo ejercicio

Consulta:

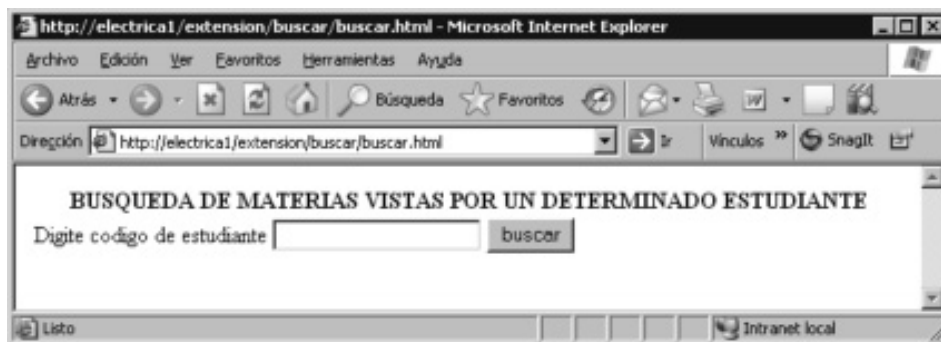
```
Select not20.codest, not20.valor, estud20.nomest, mat20.codmat, mat20.nommat
from not20, estud20, mat20
where not20.codest = estud20.codest
and not20.codmat = mat20.codmat
```

Resultado:

NOT20		EST20	MAT20	
codest	valor	nomest	codmat	nommat
1	4	Luis	1	Matemáticas
1	3	Luis	2	Física
2	5	Juan	2	Física
1	5	Luis	3	Religión

Aplicación PHP

Queremos hacer una página que permita encontrar las materias que está viendo determinado estudiante, y pensamos que la siguiente interfaz de usuario estaría bien:



El código fuente de esta página HTML es el siguiente:

```
<html>
<center>
<strong>BUSQUEDA DE MATERIAS VISTAS POR UN DETERMINADO
ESTUDIANTE</strong>
</center>
```

```
<table>
<form name=formal action=buscar.php method=get>
<tr>
<td>
Digite codigo de estudiante
</td>
<td>
<input type=text name=codigo>
</td>
<td>
<input type=submit value=buscar>
</td>
</tr>
</form>
</table>
</html>
```

Ahora realizamos la página PHP que es capaz de ejecutar la consulta y mostrar las diversas materias:

```
<?php
$conecccion = mysql_connect("localhost","extension",123);
mysql_select_db("extension",$conecccion);

$codigo1 = $_GET["codigo"];
echo "El codigo es ".$codigo1;

$consulta = "Select not20.codest, ";
$consulta = $consulta."not20.valor, ";
$consulta = $consulta. "estud20.nomest, ";
$consulta = $consulta. "mat20.codmat, ";
$consulta = $consulta. "mat20.nommat ";
$consulta = $consulta. "from not20, estud20, mat20 ";
$consulta = $consulta." where not20.codest = estud20.codest ";
$consulta = $consulta. "and not20.codmat = mat20.codmat and not20.codest=
".$codigo1;
$consulta = $consulta. ";;";

$resultado = mysql_query($consulta);
```



```

echo "<table border=6>";
echo "<tr>";

echo "<td>CODIGO ESTUDIANTE";
echo "</td>";

echo "<td>NOMBRE ESTUDIANTE";
echo "</td>";

echo "<td>CODIGO MATERIA";
echo "</td>";

echo "<td>NOMBRE MATERIA";
echo "</td>";

echo "<td>VALOR NOTA";
echo "</td>";

echo "</tr>";

while($resul = mysql_fetch_array($resultado)){
echo "<tr>";

echo "<td>";
echo $resul["codest"];
echo "</td>";

echo "<td>";
echo $resul["nomest"];
echo "</td>";

echo "<td>";
echo $resul["codmat"];
echo "</td>";

echo "<td>";
echo $resul["nommat"];
echo "</td>";

```

```
echo " <td>";  
echo $resul["valor"];  
echo " </td>";  
  
echo "</tr>";  
}  
echo "</table>";  
  
?>
```

Inserción de datos validados _____

Objetivo general

El objetivo general de este ejercicio es realizar inserciones en una base de datos, pero validando que sean coherentes antes de enviarlos a la base de datos.

Objetivo específico

- Crear las tablas en la base de datos para el ejercicio de búsqueda.
- Insertar los registros necesarios en las tablas por medio de PHP.
- Realizar dos ejercicios de consultas para manejar consultas de inserción.
- Comprender cómo hacer validaciones con JavaScript.
- Desarrollar la aplicación PHP que permita la inserción de registros validados.

Metodología

Básicamente, de 8 a 10 a. m. se expone en el tablero la secuencia de los temas que se van a realizar en el curso. De 10 a. m. a 12 m. se exponen los fundamentos del lenguaje HTML en el tablero. De 1 a 3 p. m. se hace un ejercicio estilo conferencia para que los estudiantes lo puedan ver con Netmeeting y se resuelven las dudas sobre dicho ejercicio. De 3 a 5 p. m. se deja un ejercicio tipo taller para que los estudiantes lo desarrollen.

Recursos

Para la parte inicial, correspondiente a la explicación, se requiere un tablero. Para la parte práctica, se requiere como mínimo una sala de informática donde el número de estudiantes por computador sea máximo dos.

Bibliografía

- Piattini, M. (2006). *Tecnología y diseño de bases de datos*. Madrid: RA-MA.

- Pons, O. *et al.* (2008). *Introducción a los sistemas de bases de datos*. Madrid: Paraninfo.
- Silberschatz, A.; Korth, H. F. y Sudarshan, S. (2006). *Fundamentos de bases de datos* (5ª edición). Madrid: McGraw-Hill/Interamericana de España, S. A. U.

Desarrollo de la clase

El siguiente ejercicio muestra cómo hacer inserciones de datos en PHP, de tal forma que si un estudiante que existe se intenta ingresar, el sistema no lo permitirá. Si el usuario intenta agregar un estudiante con nombre vacío, tampoco lo deja insertar, y finalmente, si el estudiante no existe y se ingresan todos los campos, el sistema lo deja ingresar.

En cada uno de los casos, el sistema informa que se hizo.

El código fuente del ejercicio es el siguiente:

```
<?php
$nombre=$_GET["nombre"];
$codigo=$_GET["codigo"];
$objcon = mysql_connect("localhost","root","");
mysql_select_db("mysql",$objcon);
$cad1 = "select * from estudiante where codigo = $codigo;";
$resultados = mysql_query($cad1);
if($resultados!=null){
    $contador = 1;
    while($resultados=mysql_fetch_array($resultados)){
        $contador = $contador + 1;
    }
    if($contador>1){
        Header("Location: http://localhost/xampp/pagina0.php?m=YaExiste");
    }
    else{
        if($nombre!=""){
            mysql_query("insert into estudiante values($codigo,$nombre);");
            Header("Location: http://localhost/xampp/pagina0.
php?m=IngresadosSusDatos");
        }
        else{
            Header("Location: http://localhost/xampp/pagina0.php?m=NombreVacio");
        }
    }
}
```

```
}
?>
```

La página de la interfaz grafica es:

```
<!--
```

Tema: Hacer una página que verifique si se puede insertar o si no se puede insertar, teniendo en cuenta que el código no este repetido.

```
-->
```

```
<?php
```

```
$mensaje = $_GET["m"];
```

```
?>
```

```
<html>
```

```
<body>
```

```
<form action=pagina1.php>
```

```
<table>
```

```
<tr>
```

```
<td>
```

```
Nombre
```

```
</td>
```

```
<td>
```

```
<input type=text name=nombre>
```

```
</td>
```

```
</tr>
```

```
<tr>
```

```
<td>
```

```
Codigo
```

```
</td>
```

```
<td>
```

```
<input type=text name=codigo>
```

```
</td>
```

```
</tr>
```

```
<tr>
```

```
<td>
```

```
<input type=reset name=can value=cancelar>
</td>
<td>
<input type=submit name=ing value=ingresar>
</td>
</tr>
<?php echo $mensaje; ?>

</table>
</form>
</body>
</html>
```

Manejo de archivos

Objetivo general

Leer, crear y escribir en archivos planos desde aplicaciones PHP, así como incluirlos en las aplicaciones PHP.

Objetivo específico

- Manejar las funciones de PHP para lectura de archivos.
- Manejar las funciones de PHP para creación de archivos.
- Manejar las funciones de PHP para escritura en archivos.
- Incluir archivos usando `required`.
- Incluir archivos usando `include`.

Metodología

Básicamente de 8 a 10 a. m. se expone en el tablero la secuencia de los temas que se van a realizar en el curso. De 10 a. m. a 12 m. se exponen los fundamentos del lenguaje HTML en el tablero. De 1 a 3 p. m. se hace un ejercicio estilo conferencia para que los estudiantes lo puedan ver con Netmeeting y se resuelven las dudas sobre dicho ejercicio. De 3 a 5 p. m. se deja un ejercicio tipo taller para que los estudiantes lo desarrollen.

Recursos

Para la parte inicial, correspondiente a la explicación, se requiere un tablero. Para la parte práctica se requiere como mínimo de una sala de informática donde el número de estudiantes por computador puede ser máximo dos.

Bibliografía

- Charle Ojeda, F. (2005). *PHP 5. Guía práctica para usuarios*. Madrid: Anaya Multimedia.

- Pons O, Miranda A, Carrillo S. (2005). *Introducción a las bases de datos: el modelo relacional*. Madrid: Thomson Paraninfo.
- Villapececellín Cid, M. (2005). *Arquitecturas de red multicapa: conexión de bases de datos*. Barcelona: Ra-Ma.

Uso de require()

La instrucción “require()” se utiliza para incluir un archivo dentro de una página PHP. Es posible que se incluya cualquier tipo de archivo, que será incluido en forma plana. En otras palabras, un documento de Microsoft Word se verá un poco diferente a como se ve con dicho procesador de texto y obviamente este tipo de cosas no son muy frecuentes en internet.

Es importante notar que el archivo incluido con la instrucción “require()”, justo antes de ser incluido el servidor web deja el modo PHP, con lo cual si alguien quiere incluir un archivo PHP y que este sea tratado por el servidor como tal, deberá colocar las etiquetas de comienzo y de finalización que indican que es código PHP.

Para hacer más práctico el aprendizaje del lector, a continuación se muestran dos páginas PHP: una primera denominada “requiere.php”, la cual utiliza la función “require()” para incluir un archivo denominado “saluda.php”, que muestra un mensaje sencillo. Veamos la página “requiere.php”:

```
<HTML>
<?php
echo "<TABLE BORDER=2>\n";
echo "<TR>\n";
echo "<TD>\n";
echo "Uso de require()\n";
echo "</TD>\n";
echo "</TR>\n";
require("saluda.php");
echo "</TABLE>\n";
?>
</HTML>
```

El contenido de la página “saluda.php” es el siguiente:

```
<?php
echo "<TR>\n";
echo "<TD>\n";
echo "Muy buenos días\n";
```



```

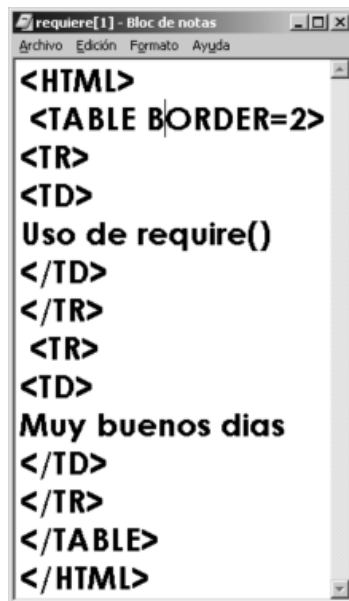
echo "</TD>\n";
echo "</TR>\n";
?>

```

Ahora bien, si montamos estas dos páginas en el servidor web y las observamos desde el *browser*, obtenemos un resultado similar al siguiente:



y si por pura casualidad observamos el código fuente desde el explorador web veremos el siguiente resultado:



La instrucción "require()" siempre lee el archivo.

Uso de include()

La instrucción “include()” es otra forma de incluir el contenido de un archivo en una página PHP, pero la diferencia radical con la sentencia “require()” radica en que “include()” se evalúa cada vez que es ejecutada, mientras que la instrucción “require()” es evaluada así esté en una condición que nunca se llega a ejecutar.

Si ejecutamos el ejercicio anterior, pero con la instrucción “include()”, es decir con las siguientes páginas:

```
<HTML>
<?php
echo "<TABLE BORDER=2>\n";
echo "<TR>\n";
echo "<TD>\n";
echo "Uso de require()\n";
echo "</TD>\n";
echo "</TR>\n";
include("saluda.php");
echo "</TABLE>\n";
?>
</HTML>
```

y el contenido de la página “saluda.php” es el siguiente:

```
<?php
echo "<TR>\n";
echo "<TD>\n";
echo "Muy buenos dias\n";
echo "</TD>\n";
echo "</TR>\n";
?>
```

Obtenemos un resultado igual, es decir, el resultado es el siguiente:



Como hemos visto, no todas las veces da igual colocar “include()” que colocar “require()”.

Manipulación de archivos

Leyendo la primera línea de un archivo

En el siguiente ejercicio se mostrará cómo hacer lecturas de archivos desde una aplicación PHP. Para ello usamos la función “fopen()”, la cual tiene dos parámetros que son:

- El nombre del archivo.
- El tipo de acceso al archivo.

La siguiente instrucción

```
fopen(“archivo.txt”, “r”);
```

indica que se abrirá un archivo llamado “archivo.txt” como solo lectura. En el caso de nuestro ejemplo, el archivo “archivo.txt” contiene:

```
ESTE ES UN LINDO
Y HERMOSO ARCHIVO
QUE VA A SER LEÍDO
POR UNA PAGINILLA PHP
ELABORADA POR EL PROFESOR
LUIS FELIPE WANUMEN SILVA
```

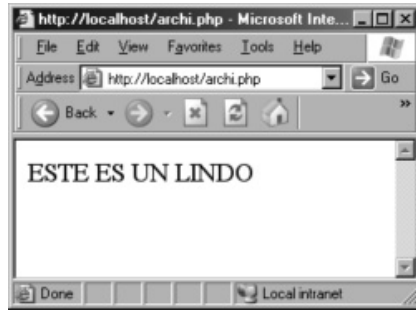
La página PHP que lee el archivo, en nuestro caso se llama “archi.php” y contiene el siguiente código fuente:

```
<?php
/*
Autor: Luis Felipe Wanumen Silva
*/

$ap_archivo = fopen(“archivo.txt”, “r”);
$primera_linea1 = fgets($ap_archivo);
echo $primera_linea1;
?>
```

Aquí podemos apreciar que la función “fopen()” devuelve un puntero al archivo, el cual en este caso se llama “\$ap_archivo”. Ahora bien, al usar dicho puntero llama-

mos la función “fgets()”, que devuelve una cadena de string con la primera línea, la cual imprimimos, produciendo una salida similar a la mostrada a continuación:

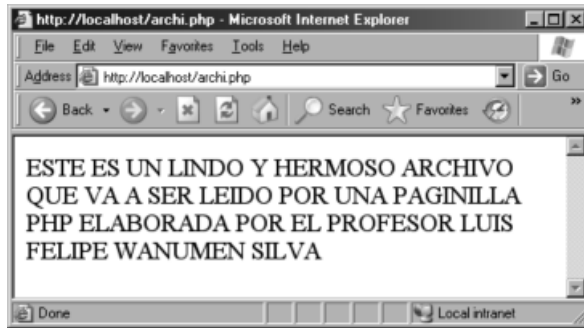


Pero, ¿no debería haber salido todo el texto de “archivo.txt”? La respuesta es que estamos usando esta función para devolver la primera línea del archivo, y si queremos que nos devuelva todo el contenido tendremos que modificar el archivo “archi.php” de la siguiente manera:

```
<?php
/*
Autor: Luis Felipe Wanumen Silva
*/

$ap_archivo = fopen(“archivo.txt”, “r”);
$primera_linea1 = fgets($ap_archivo);
echo $primera_linea1;
while($primera_linea1){
    $primera_linea1 = fgets($ap_archivo);
    echo $primera_linea1;
}
?>
```

Lo cual imprime inicialmente la primera línea de “archivo.txt”, y a continuación, en forma cíclica, continúa imprimiendo las demás líneas simplemente llamando de nuevo la función “fgets()”. El resultado que se produce al ejecutar la anterior página PHP modificada es el siguiente:



Es importante notar que los saltos de página no se toman, debido a que esto es código HTML. Si en el fuente no existe la instrucción “
”, no existe salto de página en el navegador. Modifiquémoslo para que se vea tal cual está en el archivo “archivo.txt”:

```
<?php
/*
Autor: Luis Felipe Wanumen Silva
*/

$ap_archivo = fopen("archivo.txt","r");
$primera_linea1 = fgets($ap_archivo);
echo $primera_linea1."<br>";
while($primera_linea1){
$primera_linea1 = fgets($ap_archivo);
echo $primera_linea1."<br>";
}
?>
```

El resultado de ejecutar la anterior página es similar al siguiente:



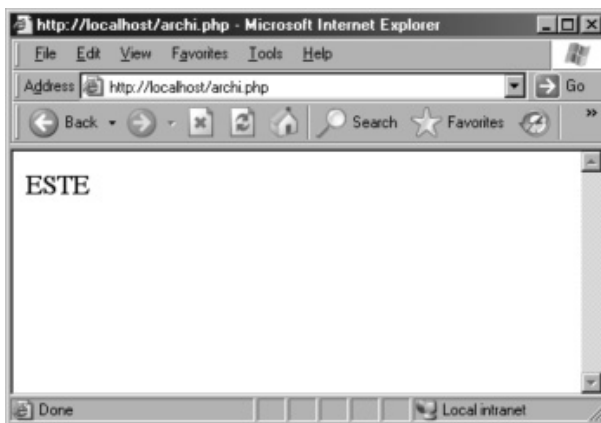
Leyendo archivos en PHP usando fread

Una forma interesante de leer el anterior archivo “archivo.txt” es con la siguiente página PHP:

```
<?php
/*
Autor: Luis Felipe Wanumen Silva
*/

$ap_archivo = fopen(“archivo.txt”,“r”);
$primera_linea1 = fread($ap_archivo,5);
echo $primera_linea1.”<br>”;
while($primera_linea1){
$primera_linea1 = fread($ap_archivo);
echo $primera_linea1.”<br>”;
}
?>
```

La cual produce el siguiente resultado:



Ahora bien, le podemos modificar el parámetro que le enviamos a la función “fread()” y ahora le enviamos el número 6:

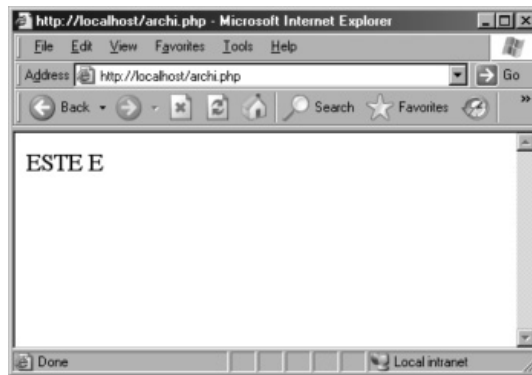
```
<?php
/*
Autor: Luis Felipe Wanumen Silva
*/
```

```

$ap_archivo = fopen("archivo.txt","r");
$primera_linea1 = fread($ap_archivo,6);
echo $primera_linea1."<br>";
while($primera_linea1){
$primera_linea1 = fread($ap_archivo);
echo $primera_linea1."<br>";
}
?>

```

Y obtenemos el siguiente resultado:



Y si usamos la instrucción:

```

$primera_linea1 = fread($ap_archivo,7);

```



Y como resultado observamos que el número pasado como parámetro indica el número de caracteres del archivo que se desean mostrar.

Al usar la instrucción:

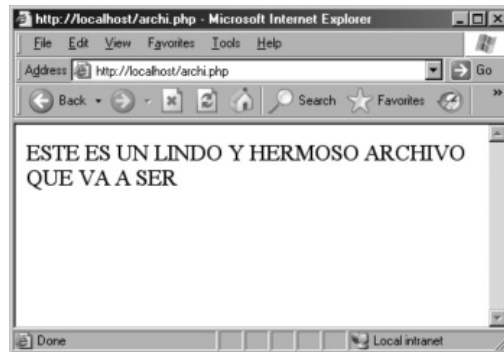
```
$primera_linea1 = fread($ap_archivo,50);
```

Donde el código fuente queda:

```
<?php
/*
Autor: Luis Felipe Wanumen Silva
*/

$ap_archivo = fopen("archivo.txt","r");
$primera_linea1 = fread($ap_archivo,50);
echo $primera_linea1."<br>";
while($primera_linea1){
$primera_linea1 = fread($ap_archivo);
echo $primera_linea1."<br>";
}
?>
```

Obtenemos el resultado:



Si colocamos un número grande, PHP lo que hace es mostrar hasta el número máximo de caracteres que tenga el archivo, por ejemplo el siguiente programa:

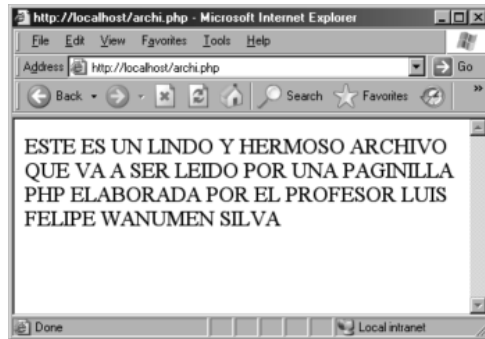
```
<?php
/*
Autor: Luis Felipe Wanumen Silva
*/

$ap_archivo = fopen("archivo.txt","r");
$primera_linea1 = fread($ap_archivo,900);
```



```
echo $primera_linea1."<br>";  
while($primera_linea1){  
$primera_linea1 = fread($ap_archivo);  
echo $primera_linea1."<br>";  
}  
?>
```

Muestra una pantalla similar a la siguiente:



Cookies con PHP

Objetivo general

El objetivo general de este ejercicio es crear y comprender el manejo de las *cookies* en PHP.

Objetivos específicos

- Comprender la teoría sobre *cookies*.
- Conocer las instrucciones de PHP para crear *cookies*.
- Aplicar el manejo de *cookies* en una aplicación de login.

Metodología

Básicamente, de 8 a 10 a. m. se expone en el tablero la secuencia de los temas que se van a realizar en el curso. De 10 a. m. a 12 m. se exponen los fundamentos del lenguaje HTML en el tablero. De 1 a 3 p. m. se hace un ejercicio estilo conferencia para que los estudiantes lo puedan ver con Netmeeting y se resuelven las dudas sobre dicho ejercicio. De 3 a 5 p. m. se deja un ejercicio tipo taller para que los estudiantes lo desarrollen.

Recursos

Para la parte inicial, correspondiente a la explicación, se requiere un tablero. Para la parte práctica, se requiere como mínimo una sala de informática donde el número de estudiantes por computador puede ser máximo dos.

Bibliografía

- Schafer, S. M. (2010). *HTML, XHTML y CSS*. Ciudad de México: Anaya Multimedia.
- Welling, L y Thomson, L. (2011). *PHP and MySQL web development*. Indiana: Sams Publishing.

Conceptos básicos de *cookies*

La siguiente página crea una *cookie*:

```
<?php  
  
$cookie_vida = 3*24*3600;  
setcookie("Clave1","111",time()+$cookie_vida);  
  
?>
```

Si ejecutas la página no ves nada, pero en realidad si ocurren cosas.



El servidor acaba de enviarle al cliente una *cookie*, cuyo nombre es “Clave1” y cuyo valor es “111”.

Ahora vamos a crear otra página PHP, que es capaz de leer el valor almacenado en la clave “Clave1” del lado del cliente.

```
<?php  
echo "la clave es" .$_COOKIE["Clave1"];  
?>
```

Al ejecutar la página aparece una interfaz similar a la siguiente:

Crear y recuperar *cookies* sencillas

La sintaxis para crear una *cookie* es con la instrucción

```
setcookie("nombre_cookie", valor_de_la_cookie);
```

Donde al primer parámetro es el nombre de la *cookie* y el segundo es el valor que se le va a dar.

Las *cookies* son archivos que se envían al cliente con el ánimo de ser recuperados en el momento en el que se necesiten. Con frecuencia las *cookies* son utilizadas para almacenar información del cliente, a veces incluso las contraseñas y los valores de

personalización de la página del cliente. Es importante tener en cuenta que para lograr que el cliente pueda almacenar *cookies* es necesario que la opción de “deshabilitar *cookies*” en el navegador no esté activada.

Para lograr comprender mejor el ejercicio que se muestra a continuación, es necesario anotar que la instrucción

```
header("Location: mostrar.php");
```

hace que el navegador se desplace hasta la página “mostrar.php”.

Nuestro ejercicio sobre el uso de *cookies* comienza con la siguiente página PHP:

```
clase4.php
<HTML>
<BODY>
<TABLE>
<FORM ACTION="cookie.php" METHOD="GET">

<TR>
<TD>
Digite Cédula:
</TD>
<TD>
<INPUT TYPE="text" NAME="cedula">
</TD>
</TR>

<TR>
<TD>
Digite Nombre:
</TD>
<TD>
<INPUT TYPE="text" NAME="nombre">
</TD>
</TR>

<TR>
<TD>
```

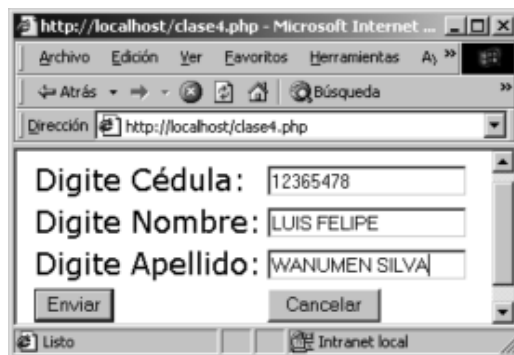
```
Digite Apellido:
</TD>
<TD>
<INPUT TYPE="text" NAME="apellido">
</TD>
</TR>

<TR>
<TD>
<INPUT TYPE="submit" VALUE="Enviar">
</TD>
<TD>
<INPUT TYPE="reset" VALUE="Cancelar">
</TD>
</TR>

</FORM>
</TABLE>

</BODY>
</HTML>
```

Esta página presenta tres cuadros de texto, en los cuales se supone que el usuario ingresa el número de cédula, el nombre y el apellido de una persona. En este caso vamos a suponer que se ingresa el nombre del autor del manual, tal como se muestra a continuación:



Cuando se presiona el botón “Enviar”, el servidor procesa la página denominada “cookie.php”, puesto que en el formulario se había predeterminado esta acción con el método GET. Para recordarlo, podemos ver que esto se hacía con la instrucción

```
<FORM ACTION="cookie.php" METHOD="GET">
```

Veamos el código de la página denominada “cookie.php”:

```
cookie.php
<?php
setcookie("cedulita", $cedula);
setcookie("nombrecito", $nombre);
setcookie("apellidito", $apellido);

header("Location: mostrar.php");
?>
```

En otras palabras, podemos observar que lo que la página hace es crear tres *cookies*, y en cada una de ellas almacena uno de los valores ingresados por el usuario en el anterior formulario.

Antes de finalizar el procesamiento de la página “cookie.php”, el servidor encuentra la función “header()”, la cual, como mencionábamos al principio de este ejercicio, sirve para desplazarse hasta una página, con lo que el servidor sabe que debe desplazarse hasta la página denominada “mostrar.php”. Veamos el código de esta página:

```
mostrar.php
<HTML>
<BODY>
<CENTER>
<TABLE BORDER = 3>

<?php
echo '<TR><TD>';
echo 'Primera cookie Cedula = '.$cedulita;
echo '</TD></TR>';

echo '<TR><TD>';
echo 'Segunda cookie Nombrecito = '.$nombrecito;
echo '</TD></TR>';

echo '<TR><TD>';
echo 'Tercera cookie Apellidito = '.$apellidito;
echo '</TD></TR>';
```

```
?>
```

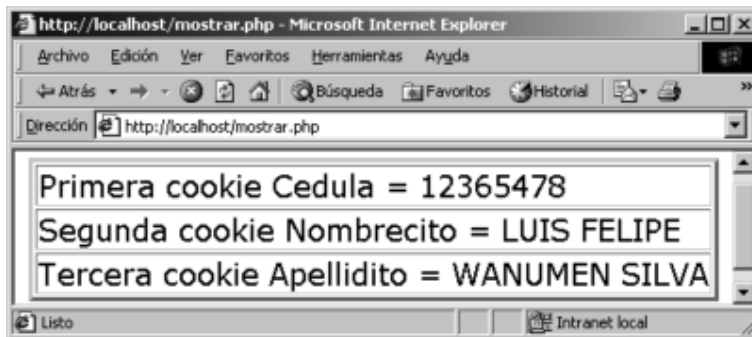
```
</TABLE>
```

```
</CENTER>
```

```
</BODY>
```

```
<HTML>
```

Como el lector puede observar, la página anterior se encarga de mostrar en una tabla los valores de las tres *cookies* anteriormente almacenadas. Podemos ver que al montar estas páginas en un servidor que permita correr páginas PHP, obtendríamos un resultado final similar al siguiente:



Usar la *cookie* para un login

La siguiente página muestra el formulario de entrada a un sitio:

```
<html>
<center>
<table>
<form action="http://127.0.0.1/establecer.php">
<tr>
<td>Digite nombre de usuario</td>
<td><input type=text name=caja1></td>
</tr>

<tr>
<td>Digite su clave</td>
<td><input type=text name=caja2></td>
</tr>

<tr>
```



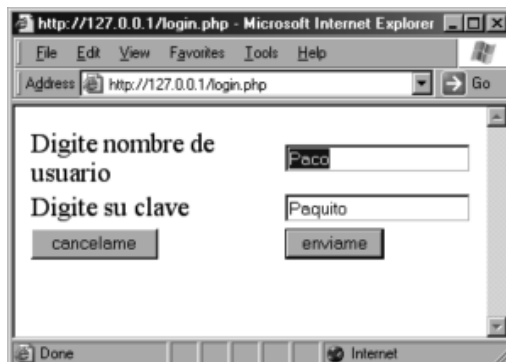
```

<td><input type=reset value=cancelame></td>
<td><input type=submit value=enviame></td>
</tr>
</form>
</table>

</center>
</html>

```

Su interfaz es similar a la siguiente:



Cuando el usuario digita la clave y el nombre de usuario, y posteriormente hace clic sobre el botón “Envíame”, se ejecuta la siguiente página:

```

<?php

$cookie_vida = 3*24*3600;
setcookie($caja1,$caja2,time()+$cookie_vida);
header("Location:perdido.php");
?>

```

La cual crea la *cookie* y reenvía la petición hacia la siguiente página:

```

<?php
if($_COOKIE["Paco"]=="Paquito"){
echo "Nombre y clave de usuario correcta";
}
else{
echo "Nombre de usuario o clave no valida";
header("Location:login.php");
}

```

```
}
```

```
?>
```

Esta página verifica que el nombre de usuario sea “Paco” y que la clave sea “Paquito”. Si se digitan datos incorrectos, se reenvía a la página “login.php”, y en caso de que sean correctos, se muestra un mensaje de bienvenida diciendo que el nombre y clave son correctos.

Programas de elementos de interfaz gráfica

En este capítulo se muestran ejemplos de programas que aprovechan las funcionalidades del lenguaje PHP mezclado con el lenguaje HTML, para mostrar elementos de interfaz gráfica HTML que interactúan entre sí.

Un combo que depende de otro combo

En algunas ocasiones es interesante observar que se necesitan desarrollar páginas en las cuales una vez escogida una opción de un combo, se tenga otro combo que tome algunos valores dependiendo de la opción elegida en el primer combo. El anterior programa se puede realizar de la siguiente manera:

```
combo1.php
<html>
<HEAD>
<script LANGUAGE="JavaScript">
var parametro=-1;
function uno() {
top.location.href = 'combo1.php?v1='+document.form1.combo1.selectedIndex;
}
</script>
</HEAD>

<body>
<FORM NAME='form1'>
<?php

if(!($conexion=mysql_connect("localhost","lucho","123456")))
{
echo "Errores conectando con el servidor";
```

```

exit();
}
else
{
echo "Conexion correcta". "<br>";
}

if (!mysql_select_db("lucho",$conexion))
{
echo "Error conectando con la base de datos.";
echo "<br>";
exit();
}
else{
echo "Base de datos correcta";
echo "<br>";
}

$result=mysql_query("select * from chicos",$conexion);
echo "<select name='combo1' size=0 onchange=uno()>";
while($row = mysql_fetch_array($result)) {
echo "<option value=' " . $row[0] . " '>" . $row[1];
echo "</option>";
}
echo "</select>";
mysql_free_result($result);

if($v1){
$v = $v1;
$result2=mysql_query("select * from chicos2 where codigo =" . $v,$conexion);
echo "<select name='combo2' size=1>";
while($row = mysql_fetch_array($result2)) {
echo "<option value=' " . $row[0] . " '>" . $row[1];
echo "</option>";
}
}

```

```

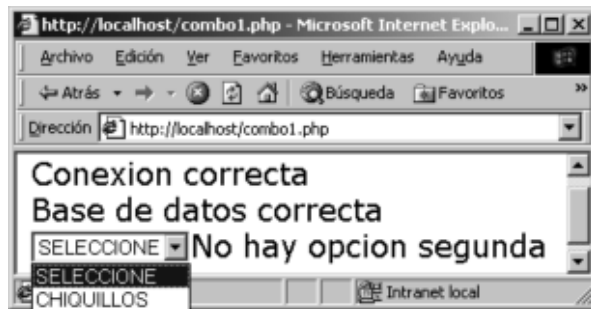
echo "</select>";
}
else
{
echo "No hay opcion segunda";
}

mysql_free_result($result2);
mysql_close($conexion);

?>
</FORM>
</body>
</html>

```

La página anterior se puede ver en el navegador de la siguiente manera:



Estamos suponiendo que el contenido de la tabla denominada “chicos” es el siguiente:

CÓDIGO	NOMBRE
0	SELECCIONE
1	CHIQUELLOS
2	NIÑOS
3	NIÑAS

y que el contenido de la tabla denominada “chicos2” es el siguiente:

CÓDIGO	NOMBRE
1	chiquillos1
1	chiquillos1
1	chiquillos2
1	chiquillos3
1	chiquillos4
1	chiquillos5
1	chiquillos6
1	chiquillos7
2	niños1
2	niños2
2	niños3
2	niños4
2	niños5
2	niños6
2	niños7
3	niñas1
3	niñas2
3	niñas3
3	niñas4
3	niñas5

Digamos que el usuario selecciona la opción “Chiquillos”. Entonces se produce el siguiente resultado gráficamente hablando:



Si el usuario selecciona la opción del primer combo denominada “Niño”, se tiene entonces el siguiente resultado gráficamente hablando:



Esto ocurre de forma similar para todas las demás opciones del primer combo. Por otra parte, es bueno que el lector note que una vez seleccionada la opción del primer combo, se logra que el segundo cambie en cuanto a su contenido. Pero ocurre algo que de pronto puede confundir al navegante, y es que en este caso el primer combo cambia, y dado que se vuelve a colocar en la pantalla, siempre aparecerá con la primera opción visible para el usuario. La corrección de este detalle se deja de tarea al lector.

Paginación en PHP

Este capítulo intenta mostrar cómo la paginación de resultados disminuye la cantidad de registros que se le entregan visualmente a un usuario final, con el fin de que no se sienta inundado con tanta información. Obviamente este tipo de técnicas son útiles cuando el número de registros en una aplicación es grande. Esta técnica no solo disminuye el número de registros en una página, sino que mejora la experiencia de usuario, al mismo tiempo que disminuye el tráfico de red.

Objetivo general

El objetivo general de este ejercicio es manejar la paginación profesional de resultados.

Objetivos específicos

- Realizar un ejercicio de paginación en PHP.
- Aplicar paginación a los problemas concretos que requieran esta solución.
- Manejar la instrucción limit en MySQL para delimitar el tamaño del conjunto de resultados.

Metodología

Básicamente, de 8 a 10 a. m. se expone en el tablero la secuencia de los temas que se van a realizar en el curso. De 10 a. m. a 12 m. se exponen los fundamentos del lenguaje HTML en el tablero. De 1 a 3 p. m. se hace un ejercicio estilo conferencia para que los estudiantes lo puedan ver con Netmeeting y se resuelven las dudas sobre dicho ejercicio. De 3 a 5 p. m. se deja un ejercicio tipo taller para que los estudiantes lo desarrollen.

Recursos

Para la parte inicial, correspondiente a la explicación, se requiere un tablero. Para la parte práctica, se requiere como mínimo una sala de informática donde el número de estudiantes por computador puede ser máximo dos.

Bibliografía

- Pons, O. *et al.* (2008). *Introducción a los sistemas de bases de datos*. Madrid: Paraninfo.
- Suehring, S.; Converse, T. y Park, J. (2009). *PHP 6 and MySQL Bible*. Indianápolis-Indiana: Wiley Pub.
- Welling, L. y Thomson, L. (2011). *PHP and Mysql Web development*. Indiana: Sams Publishing.

Desarrollo del contenido de la clase

Supongamos que tenemos una tabla llamada “modelos” en MySQL con los siguientes datos:

CÓDIGO	NOMBRE
1	Ana Sofía Henao
2	Ana Sofía Vergara
3	Natalia París
4	Claudia Bahamón
5	Claudia de Colombia
6	Tatiana Ariza
7	Sofía Roca
8	Luz Roca
9	Sofía París
10	Tatiana de los Ríos
11	Carolina Cruz
12	Andrea Serna
13	Lina Marulanda
14	Carolina Sabino
15	Paula Andrea Betancur
16	Adriana Arboleda
17	Cindy Crawford
18	Patricia Janiot
19	Paola Turbay

Supongamos que queremos mostrar de a 5 registros por página. Quedarían 4 páginas, cada una con los datos como se muestra a continuación:

Código	Nombre	Página
1	Ana Sofía Henao	1
2	Ana Sofía Vergara	
3	Natalia París	
4	Claudia Bahamón	
5	Claudia de Colombia	
6	Tatiana Ariza	2
7	Sofía Roca	
8	Luz Roca	
9	Sofía París	
10	Tatiana de los Ríos	
11	Carolina Cruz	3
12	Andrea Serna	
13	Lina Marulanda	
14	Carolina Sabino	
15	Paula Andrea Betancourt	
16	Adriana Arboleda	4
17	Cindy Crawford	
18	Patricia Janiot	
19	Paola Turbay	

Tengamos en cuenta que la instrucción

```
Select * from modelos limit 0,5
```

muestra los siguientes datos:

Código	Nombre
1	Ana Sofía Henao
2	Ana Sofía Vergara
3	Natalia París
4	Claudia Bahamón
5	Claudia de Colombia

De forma semejante, las siguientes instrucciones producen estos resultados en MySQL.

La instrucción:

```
Select * from modelos limit 1,6
```

Código	Nombre
2	Ana Sofía Vergara
3	Natalia París
4	Claudia Bahamón
5	Claudia de Colombia
6	Tatiana Ariza
7	Sofía Roca

```

C:\WINDOWS\system32\cmd.exe - mysql
mysql> select * from modelos limit 1,6;
+----+-----+
| codigo | nombre |
+----+-----+
| 2 | Ana S |
| 3 | Natal |
| 4 | Claud |
| 5 | Claud |
| 6 | Tatia |
| 7 | Sofia |
+----+-----+
6 rows in set (0.00 sec)

mysql>
  
```

La instrucción:

Select * from modelos limit 2,7

CÓDIGO	NOMBRE
3	Natalia París
4	Claudia Bahamón
5	Claudia de Colombia
6	Tatiana Ariza
7	Sofía Roca
8	Luz Roca
9	Sofía París

```

C:\WINDOWS\system32\cmd.exe - mysql
mysql> select * from modelos limit 2,7;
+----+-----+
| codigo | nombre |
+----+-----+
| 3 | Natal |
| 4 | Claud |
| 5 | Claud |
| 6 | Tatia |
| 7 | Sofia |
| 8 | Luz R |
| 9 | Sofia |
+----+-----+
7 rows in set (0.00 sec)

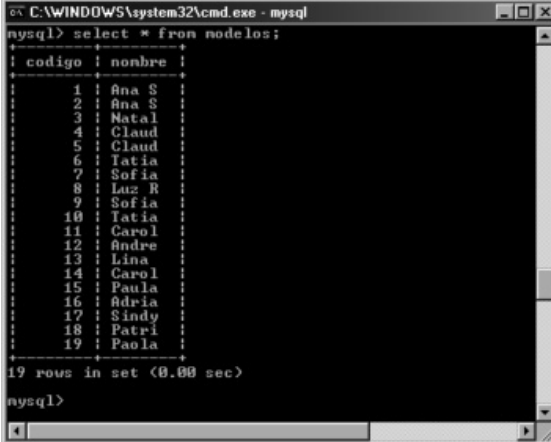
mysql> _
  
```

Vemos que la primera cifra indica el número del registro en el que comienza a mostrarse del conjunto de resultados. La segunda cifra muestra el número de registros que es necesario contar a partir del anterior registro para visualizar.

Se dejará que el lector averigüe por qué el siguiente *script*

```
insert into modelos values(1, 'Ana Sofia Henao');
insert into modelos values(2, 'Ana Sofia Vergara');
insert into modelos values(3, 'Natalia Paris');
insert into modelos values(4, 'Claudia Bahamon');
insert into modelos values(5, 'Claudia de Colombia');
insert into modelos values(6, 'Tatiana Ariza');
insert into modelos values(7, 'Sofia Roca');
insert into modelos values(8, 'Luz Roca');
insert into modelos values(9, 'Sofia Paris');
insert into modelos values(10, 'Tatiana de los Ríos');
insert into modelos values(11, 'Carolina Cruz');
insert into modelos values(12, 'Andrea Serna');
insert into modelos values(13, 'Lina Marulanda');
insert into modelos values(14, 'Carolina Sabino');
insert into modelos values(15, 'Paula Andrea Betancur');
insert into modelos values(16, 'Adriana Arboleda');
insert into modelos values(17, 'Cindy Crawford');
insert into modelos values(18, 'Patricia Janiot');
insert into modelos values(19, 'Paola Turbay');
```

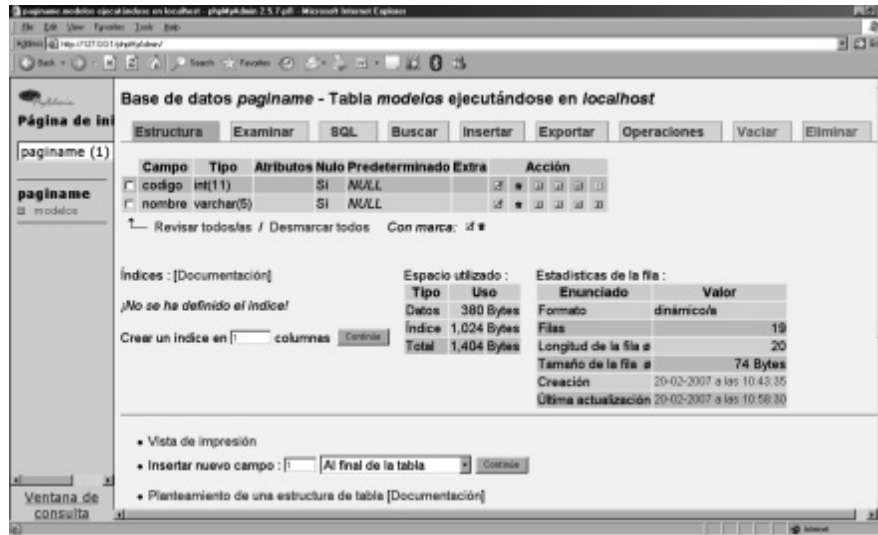
inserta los valores que a continuación se muestran:



```
C:\WINDOWS\system32\cmd.exe - mysql
mysql> select * from modelos;
+----+-----+
| codigo | nombre |
+----+-----+
| 1 | Ana S |
| 2 | Ana S |
| 3 | Natal |
| 4 | Claud |
| 5 | Claud |
| 6 | Tatia |
| 7 | Sofia |
| 8 | Luz R |
| 9 | Sofia |
| 10 | Tatia |
| 11 | Carol |
| 12 | Andre |
| 13 | Lina |
| 14 | Carol |
| 15 | Paula |
| 16 | Adria |
| 17 | Sindy |
| 18 | Patri |
| 19 | Paola |
+----+-----+
19 rows in set (0.00 sec)

mysql>
```

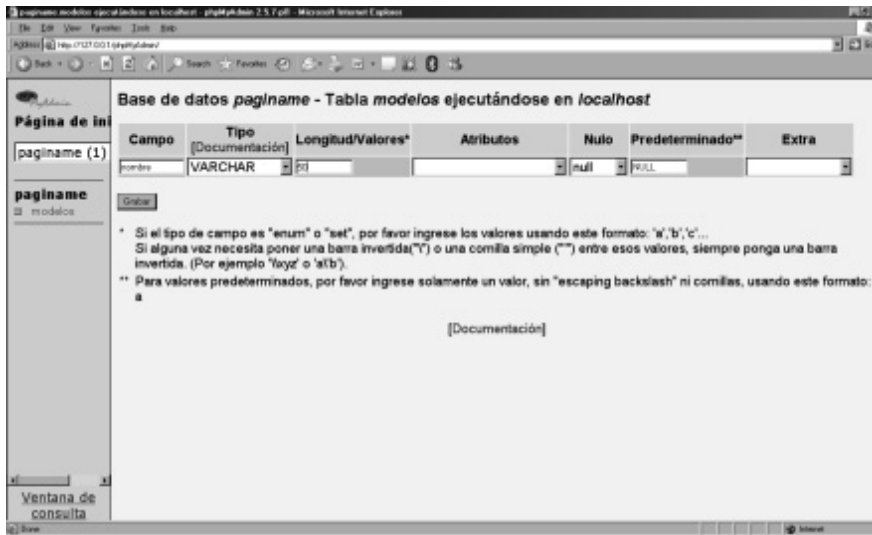
Aquí podemos apreciar que se pierden algunos caracteres. La respuesta es que el tipo de datos se ha creado con una extensión de 5 caracteres:



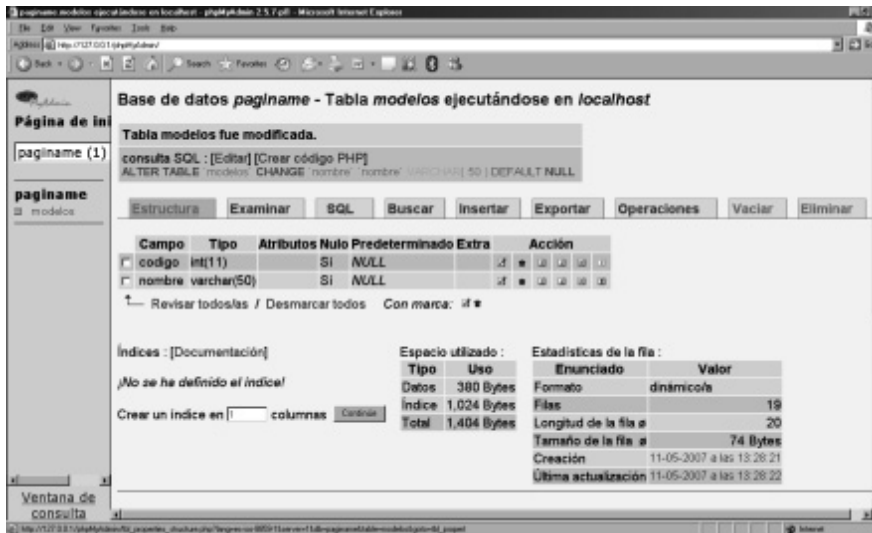
Podemos modificar la extensión del número de caracteres que se pueden almacenar en el campo “Nombre”, tal como se muestra a continuación. Hacemos clic en el ícono que tiene el lápiz de edición y que está al frente de la fila “Nombre”



Cambiamos el 5 por el 50



y después hacemos clic en grabar.



También se pudo haber modificado el tamaño del campo por consola, tal como se muestra a continuación:

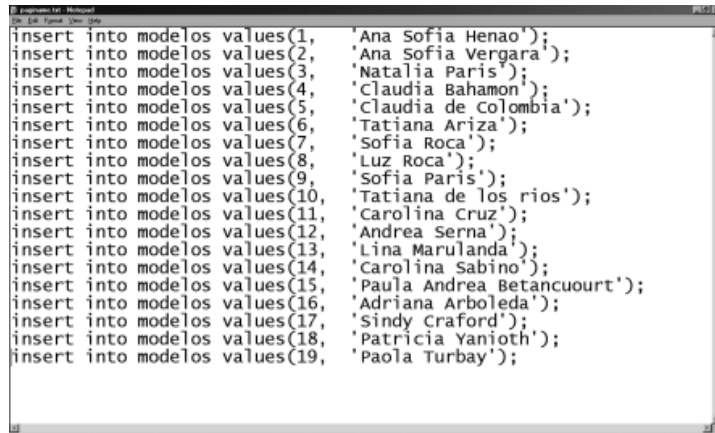
```
C:\WINDOWS\system32\cmd.exe - mysql
mysql> alter table modelos change nombre nombre varchar(50);
Query OK, 19 rows affected (0.09 sec)
Records: 19 Duplicates: 0 Warnings: 0
mysql> _
```

Que tendría el mismo efecto que la instrucción:

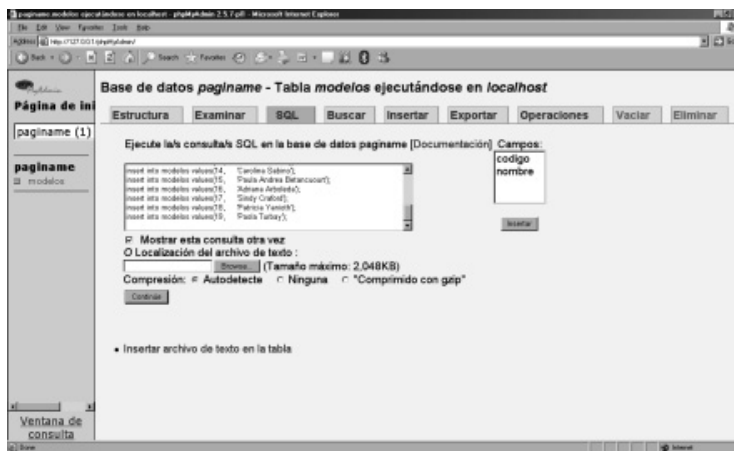


Esto admitiría campos vacíos o nulos en “Nombre” y un tamaño de hasta 50 caracteres. Observemos que si el campo no tiene una longitud suficiente para almacenar los datos que se realicen en las instrucciones “insert”, MySQL no nos informa de esto, sino que simplemente trunca los datos almacenando lo que pueda. Es necesario prever esta situación, ya que podría prestarse para errores sin que nos demos cuenta.

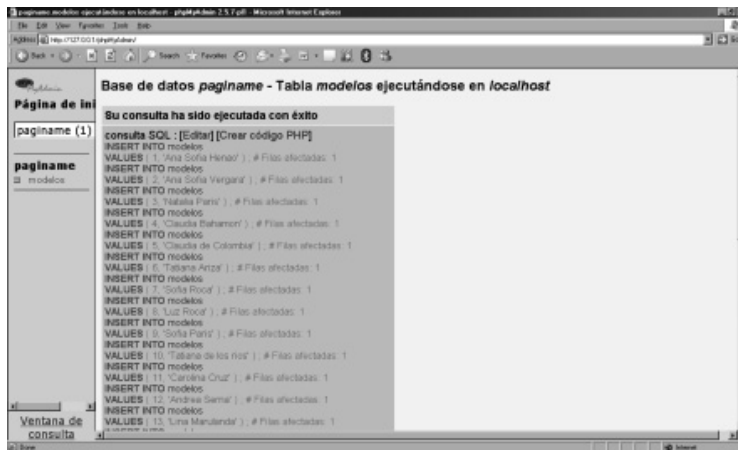
Es bueno tener en cuenta que las instrucciones de inserción también se pueden digitar en un archivo de texto



Y luego copiarlas en el administrador de MySQL tal como se muestra a continuación



para terminar ejecutándolas



Para no salirnos del ejercicio de paginación, veamos el código completo de la página que es capaz de mostrar de a 5 registros por página:

```
<?php
if($actual==null){
// echo "Eres primiparo en la pagina <br>";
$actual=1;
}
else{
// echo "Eres veterano en la pagina <br>";
}
// echo "<strong>Estoy en la pagina ".$actual."</strong><br>";

$numero_registros = 19;
$registros_por_pagina = 5;

//Cuantas paginas hay que mostrar?
$numero_paginas = ($numero_registros / $registros_por_pagina);
$division = (int)($numero_registros / $registros_por_pagina);
$division = $division * $registros_por_pagina;
if($division==$numero_registros){
// echo "La division es exacta";
}
else{
```

```

// echo "La division no es exacta <br><br><br>";
$numero_paginas++;
}
echo "<center>";
for($i=1;$i<=$numero_paginas;$i++){
echo "<a href=http://127.0.0.1/paginame/pa.php?actual=".$i.">".$i."</a>";
}
echo "</center>";

// echo "La consulta que toca hacer es <br>";
$inicio = ($actual*$registros_por_pagina) - $registros_por_pagina;
$final = ($actual*$registros_por_pagina);
// echo $numero_paginas=(int)$numero_paginas;

$consulta = "";
if($actual==$numero_paginas){
$final = $numero_registros;
$cuadre = $numero_registros - $inicio;
$consulta = "Select * from modelos limit ".$inicio.", ".$cuadre."";
} // Cierra if para el caso del patito feo
else{
$consulta = "Select * from modelos limit ".$inicio.", ".$registros_por_pagina."";
}

// echo "INICIO = ".$inicio."<br>";
// echo "FINAL = ".$final."<br>";
// echo $consulta;

if(!($conexion=mysql_connect("localhost","root","")))
{
// echo "Errores conectando con el servidor";
exit();
}
else
{
// echo "Conexion correcta"."<br>";
}

```

```

if (!mysql_select_db("paginame",$conexion))
{
// echo "Error conectando con la base de datos.";
// echo "<br>";
exit();
}
else{
// echo "Base de datos correcta";
// echo "<br>";
}
$consulta = $consulta.".";
$result=mysql_query($consulta,$conexion);

echo "<center><table border = 8>";
while($row = mysql_fetch_array($result)) {
printf("<tr><td>%s</td><td>%s</td></tr>", $row["codigo"],$row["nombr
re"]);
}
echo "</table></center>";
mysql_free_result($result);
mysql_close($link);
?>

```

El resultado de la ejecución de la anterior pagina es el siguiente:



Si el usuario hace clic en el enlace que tiene el número 2 obtenemos



Si hace clic sobre el enlace con el número 3 se obtiene:



Si se hace clic sobre el enlace con el número 4 obtenemos:



Con lo anterior podemos observar que nuestro ejercicio de paginación funciona. Para comprender el código anterior es bueno que se entienda que el programa de pagina-

ción usa unas variables “inicio” y “fin”, pero que realmente la más importante es la variable “inicio”, que especifica en qué registro comienza la búsqueda. La tabla siguiente muestra la situación:

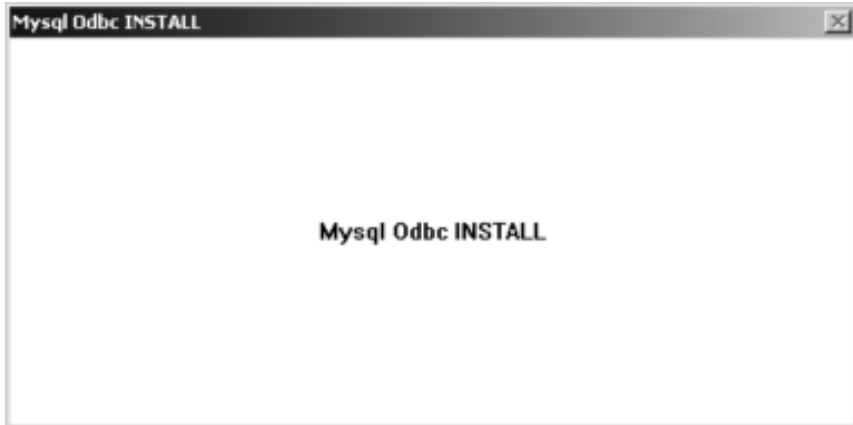
Código	Nombre	Página	Inicio	Fin	Consulta
1	Ana Sofia Henao	1	1	5	Select * from modelos limit 0,5
2	Ana Sofia Vergara		1	5	
3	Natalia París		1	5	
4	Claudia Bahamón		1	5	
5	Claudia de Colombia		1	5	
6	Tatiana Ariza	2	6	10	Select * from modelos limit 5,5
7	Sofia Roca		6	10	
8	Luz Roca		6	10	
9	Sofia París		6	10	
10	Tatiana de los Ríos		6	10	
11	Carolina Cruz	3	11	15	Select * from modelos limit 10,5
12	Andrea Serna		11	15	
13	Lina Marulanda		11	15	
14	Carolina Sabino		11	15	
15	Paula Andrea Betancur		11	15	
16	Adriana Arboleda	4	16	19	El patito feo queda para que se verifique como taller
17	Cindy Crawford		16	19	
18	Patricia Janiot		16	19	
19	Paola Turbay		16	19	

Interoperar entre MySQL y otros motores _____

La interoperabilidad entre MySQL y otros motores de bases de datos es algo que inquieta a muchas personas, sobre todo a aquellas que mantienen aplicaciones en otros motores y están apenas experimentando con MySQL. La razón para que esto suceda no es única y puede deberse a factores como la posibilidad de ver qué tan confiable es volver a tomar las aplicaciones que antes se tenían en otros motores, o ver qué tan compatible es MySQL con otros manejadores. De todas maneras, sea cual sea la razón por la que el tema interese, es fundamental que el lector tenga una idea de la importancia de interoperar entre MySQL y otros motores de bases de datos.

Instalar ODBC para MySQL

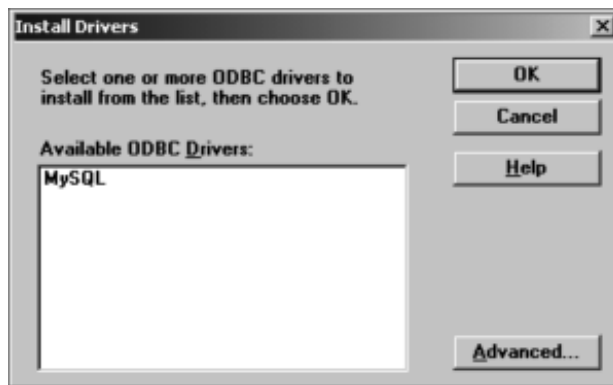
Lo primero que es necesario hacer es instalar el conector ODBC para MySQL. En este caso se podría utilizar “MyODBC”, el cual contiene un archivo denominado “setup.exe” que al ser ejecutado muestra una pantalla similar a la siguiente:



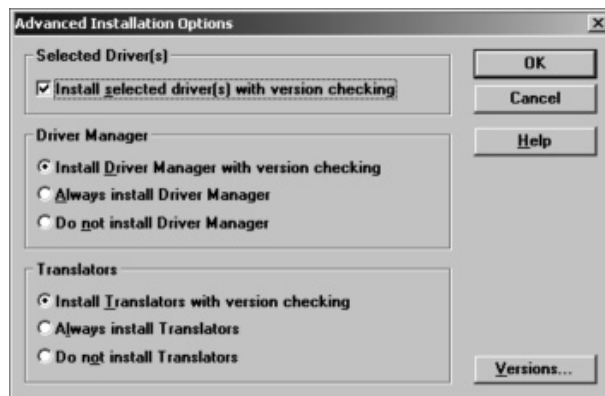
y más adelante muestra



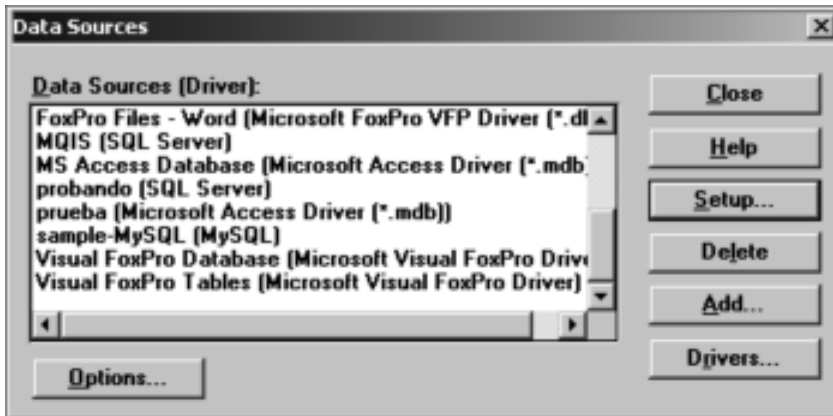
Aquí se supone que el usuario presiona el botón “Continue”, después de lo cual el instalador pide que se seleccione el driver ODBC que se desea instalar. En este caso solamente aparece disponible el driver ODBC para MySQL, tal como se muestra en la siguiente figura:



Es aconsejable instalar el driver disponible en la lista. Por ejemplo, el driver MySQL que aparece por defecto es el aconsejado para instalar.

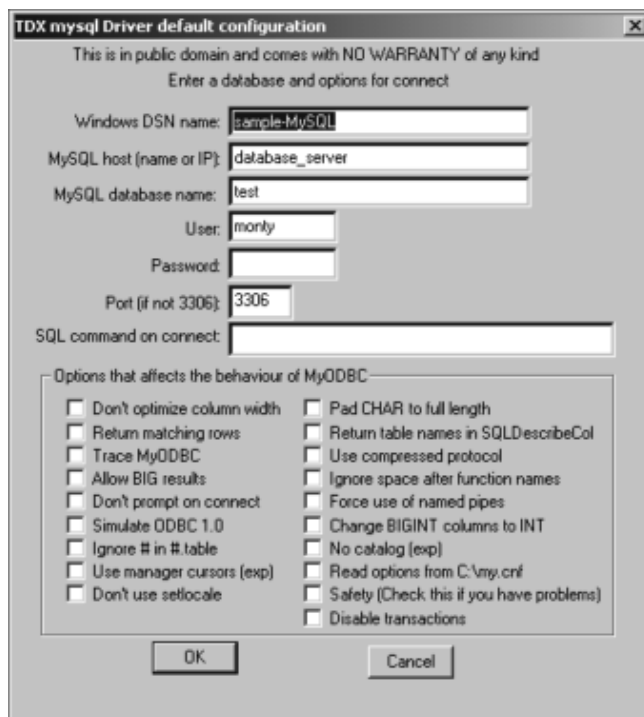


El usuario selecciona el driver que desea instalar y el asistente le presenta una pantalla similar a la siguiente:

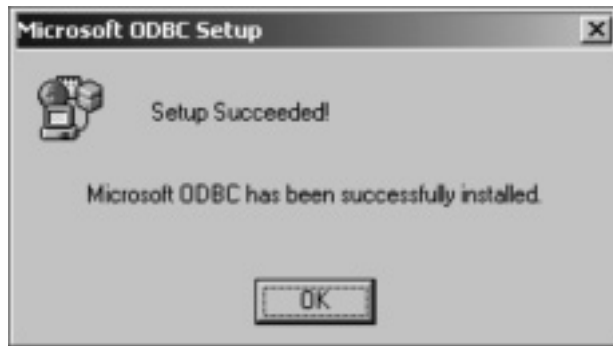


que no es más que el reflejo de los orígenes de datos instalados en el equipo. Como el amigo lector puede apreciar, en este caso el origen de datos que creó el asistente para MySQL fue uno denominado “sample-MySQL”.

Si se desea obtener más información a cerca del origen de datos instalado por el asistente, podemos hacerle doble clic en la pantalla anterior, con lo cual se nos presentará un recuadro similar a la siguiente:



El proceso de instalación del ODBC para MySQL no es más. Cuando el usuario presione el botón “OK” aparecerá una pantalla similar a la siguiente:

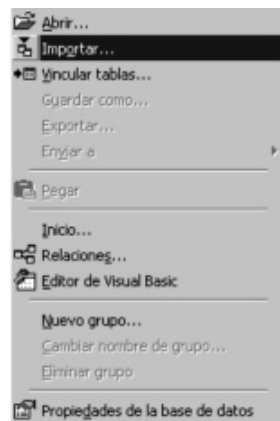


Importar datos de MySQL hasta Access

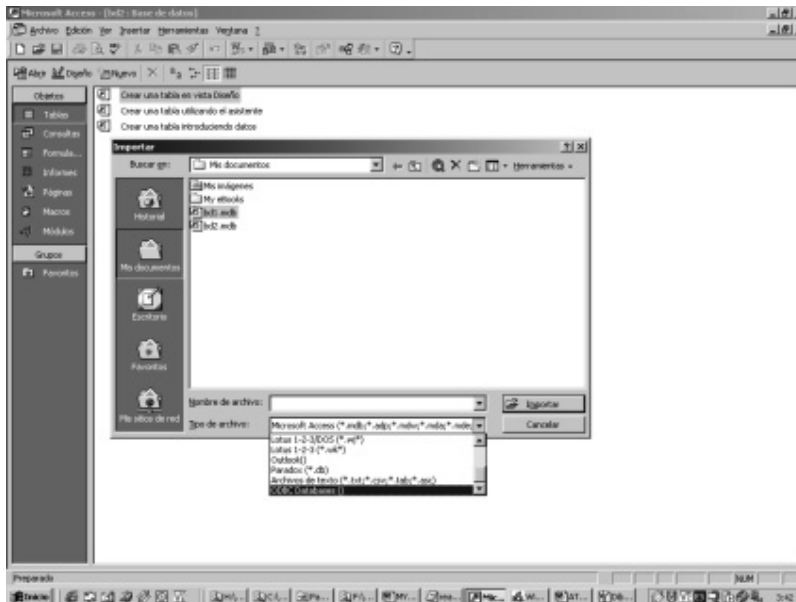
Se abre una base de datos en Access:



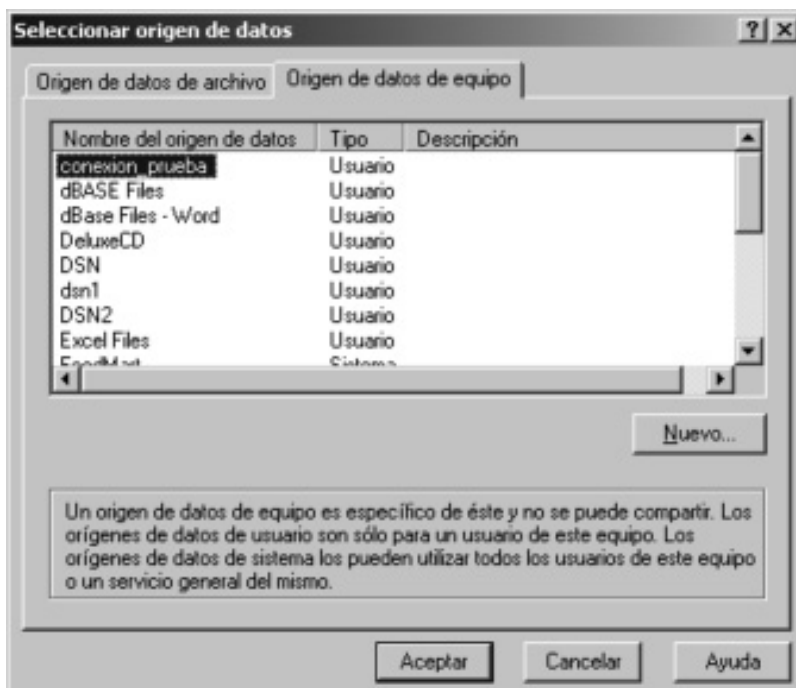
Se hace clic derecho sobre “Tablas”.



Se elige la opción “Importar”.



En la parte inferior se escoge “ODBC Databases” y se escoge el origen de datos de equipo, tal como se muestra en la siguiente figura:



Después de esto se escogen las tablas que se desean importar en forma similar a como se muestra a continuación:



Referencias

- Charte Ojeda, F. (2005). *PHP 5. Guía práctica para usuarios*. Madrid: Anaya Multimedia.
- Delisle, M. (2007). *Dominar phpMyAdmin para una administración efectiva de MySQL*. Quebec: Packt Publishing.
- Heurtel, O. (2009). *PHP y MySQL. Domine el desarrollo de un sitio web dinámico e interactivo*. Barcelona: Ediciones ENI.
- Molinero, M. (2006). *Curso de SQL*. Ciudad de México: Anaya Multimedia.
- Piattini, M. et al. (2006). *Tecnología y diseño de bases de datos*. Madrid: RA-MA.
- Pons Capote, O. et al. (2005). *Introducción a las bases de datos: el modelo relacional*. Madrid: Thomson Paraninfo.
- Pons, O. et al. (2008). *Introducción a los sistemas de bases de datos*. Madrid: Paraninfo.
- Schafer, S. M. (2010). *HTML, XHTML y CSS*. Ciudad de México: Anaya Multimedia.
- Silberschatz, A.; Korth, H. F. y Sudarshan, S. (2006). *Fundamentos de bases de datos* (5ª edición). Madrid: McGraw Hill/Interamericana de España, S. A. U. Editorial.
- Suehring, S.; Converse, T. y Park, J. (2009). *PHP 6 and MySQL Bible*. Indiana: Wiley Pub.
- Villapececlín Cid, M. (2005). *Arquitecturas de red multicapa: conexión de bases de datos*. Barcelona: Ra-Ma.
- Welling, L y Thomson, L. (2011). *PHP and MySQL web development*. Indiana: Sams Publishing.

Autores

Luis Felipe Wanumen Silva

Ingeniero de Sistemas y especialista en Ingeniería de Software, de la Universidad Distrital Francisco José de Caldas; magíster en Ingeniería de Sistemas y Computación, de la Pontificia Universidad Javeriana. Docente de Planta de la Universidad Distrital Francisco José de Caldas e integrante del grupo de investigación Metis de la misma universidad. Ha publicado más de 12 artículos de investigación y ha desarrollado aplicaciones web y móviles; tiene cursos de programación en Youtube sobre el lenguaje C y el lenguaje Java totalmente gratuitos y ha sido coach en las maratones de programación organizadas por REDIS/ACM.

Laura Ximena García Vaca

Ingeniera de Sistemas, de la Universidad de Los Andes. Monitorea académica en la misma universidad en Desarrollo Logístico y Tecnologías de Información y su aplicación para apoyar el funcionamiento de las empresas. Tiene experiencia en el área de soporte en Microsoft Colombia.

Darín Jairo Mosquera Palacios

Ingeniero de Sistemas y especialista en Teleinformática y magíster en Teleinformática, de la Universidad Distrital Francisco José de Caldas. Docente de planta de la Universidad Distrital Francisco José de Caldas y Director del grupo de investigación Orión de la Facultad Tecnológica, de la misma universidad. Ha publicado una serie de artículos de investigación en áreas como la telemática, las redes, el desarrollo de software, las metodologías de desarrollo y la seguridad informática y ha dirigido proyectos de grado y ha participado en diferentes ponencias.

Este libro se
terminó de imprimir
en agosto de 2017
en la Editorial UD
Bogotá, Colombia