

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 quitemos tiempo que muy seguramente recuperaremos.

© 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

Arrays en `php` 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
hiper enlaces 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 im-

portante. 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.

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

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 desorientaría 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:

<code><?</code>	<code>?></code>

<code><?php</code>	<code>?></code>
<code><script language="php"></code>	<code></script></code>
<code><%</code>	<code>%></code>

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 con figurar 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 Wanu men Silva”, pero observe que dicho nombre tiene unas comillas sencillas y esto tam bién 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.

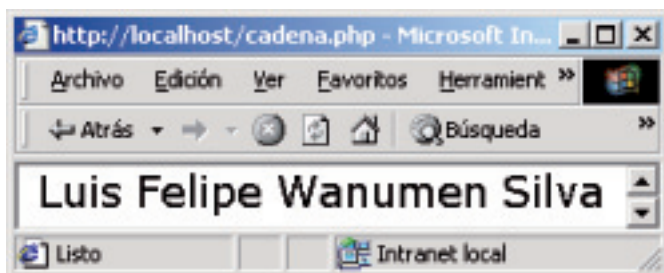
Concatenar cadenas en php

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

```
<?php
$scad1 = ' Luis ';
$scad2 = ' Felipe ';
$scad3 = ' Wanumen ';
$scad4 = ' Silva ';
$scad5 = $scad1 . $scad2 . $scad3 . $scad4;
echo $scad5;
?>
```

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 im

prime 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:

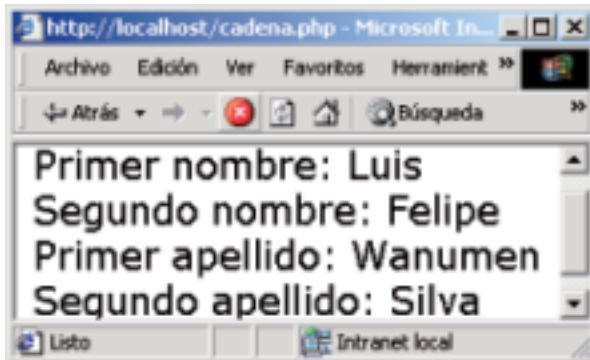
Arrays en php

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:

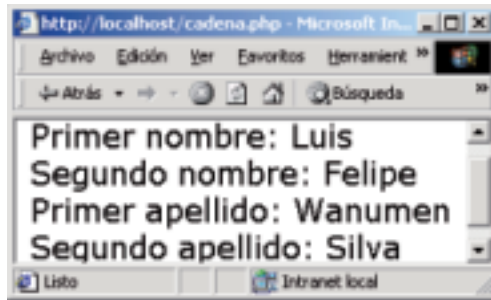


Arrays asociativos

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

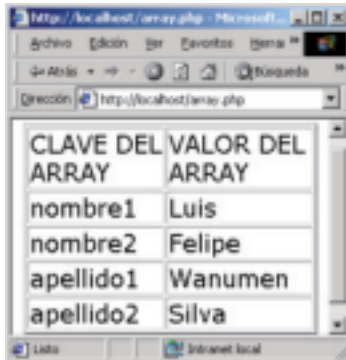
```

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



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

navegador:

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 un temano 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>";

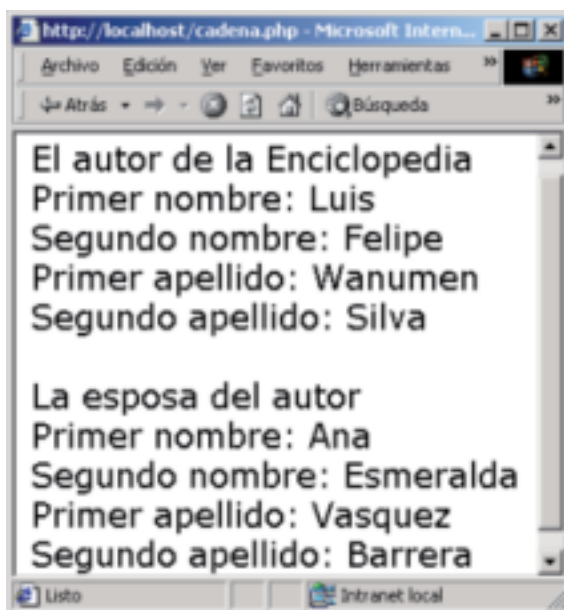
```

```

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>

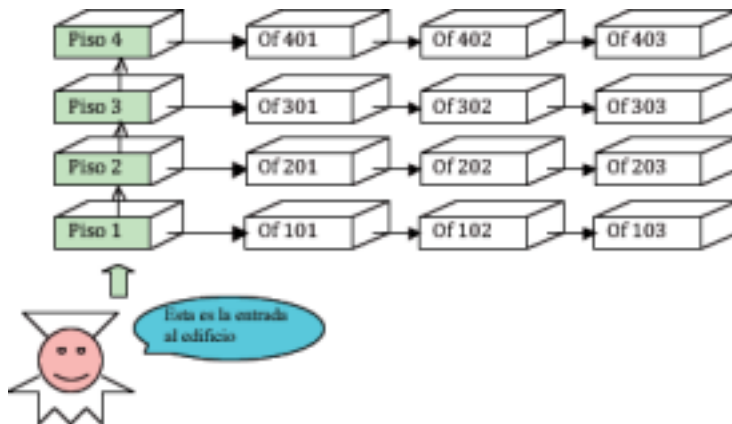
```

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* don de 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";
}
?>

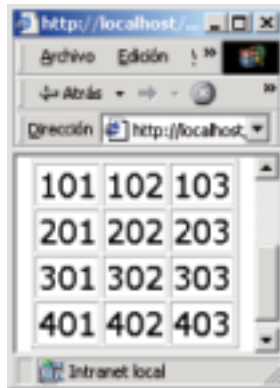
```

```

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

```

Lo anterior produce el siguiente resultado:



101	102	103
201	202	203
301	302	303
401	402	403

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";

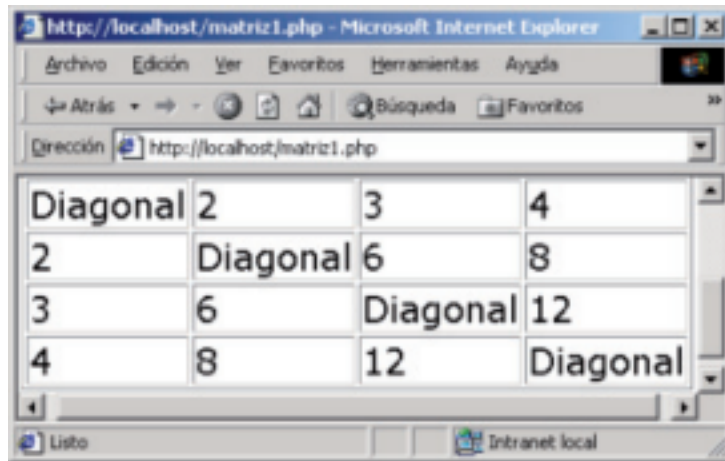
```

```

$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

31

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

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"];

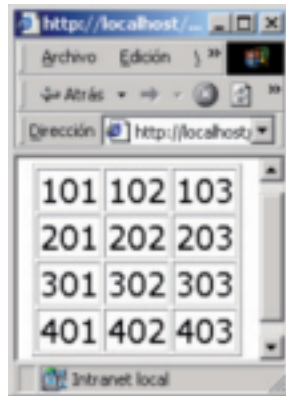
```

```

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:



A screenshot of a web browser window displaying a table with 4 rows and 3 columns. The table contains numbers from 101 to 403, arranged in a grid. The browser's address bar shows 'http://localhost/'.

101	102	103
201	202	203
301	302	303
401	402	403

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:

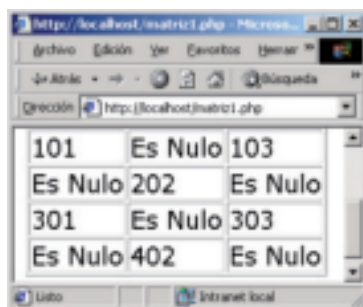
1+1=2	1+2=3	1+3=4
2+1=3	2+2=4	2+3=5
3+1=4	3+2=5	3+3=6
4+1=5	4+2=6	4+3=7

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> ';

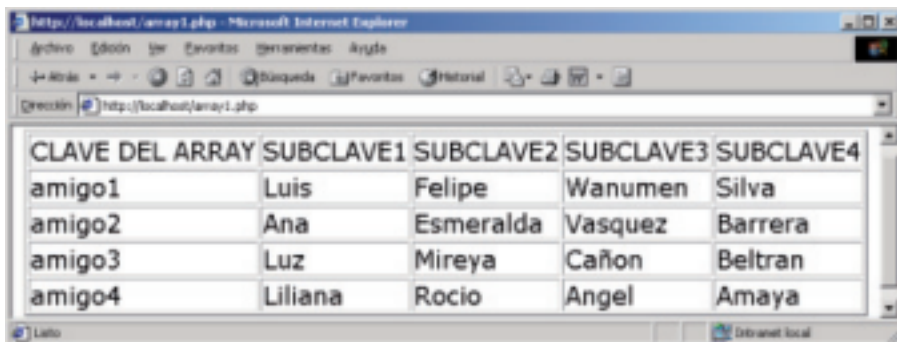
```

```

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:



41

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
Software de Microsoft.	Software 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;

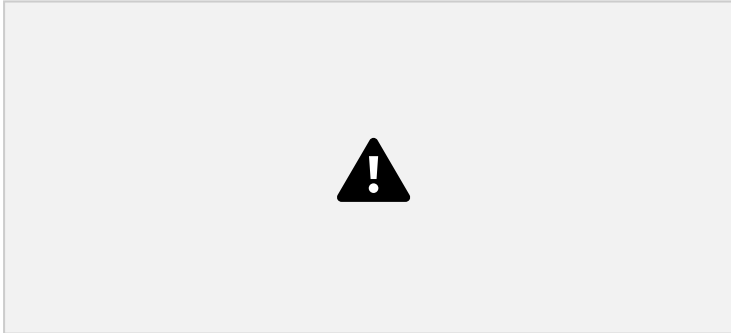
    function figura(){
        $area=1;
        $perimetro=2;
        $lados=3;
        echo 'Area= ' . $area;
        echo 'Perimetro= ' . $perimetro;
        echo 'lados= ' . $lados;
    }
}
```

```
}
```

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

```
?>
```

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



46

Manual de PHP y MySQL

Error común al inicializar variables en clases con php

Un error muy común al inicializar las variables contenidas en una clase es inicializar las 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:

47

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

```
<?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:

48

Manual de PHP y MySQL

```
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';
}

}

```

49

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

```

$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 inicia liza las variables globales de la clase e imprime sus valores.

Recordemos que la instrucción

echo ‘
 ‘;

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:

52

Manual de PHP y MySQL

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:

53

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



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...';
```

54

Manual de PHP y MySQL

```
}
```

```
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 de clarado 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

57

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

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>

59

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

<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:

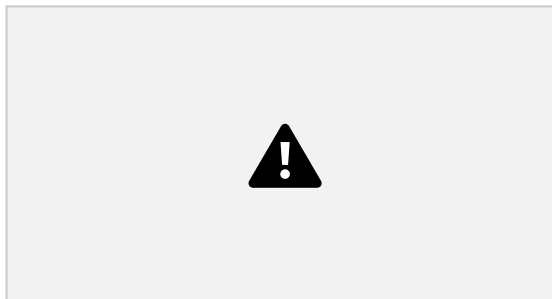
abuelo abuela

padre madre

hijo hija

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:

abuelo abuela

padre

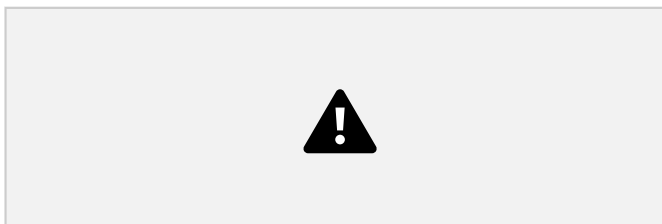
Como hemos visto, esto no está permitido en `php`, por lo menos en las versiones actuales. Algunos dicen que en próximas versiones se podrá implementar la herencia múltiple en `php`. Esperamos que así sea.

66

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:



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

67

Luego se puede conectar con esta base de datos, mediante la instrucción `Connect lucho` tal como se puede apreciar en la figura:



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:



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:



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:



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:



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.

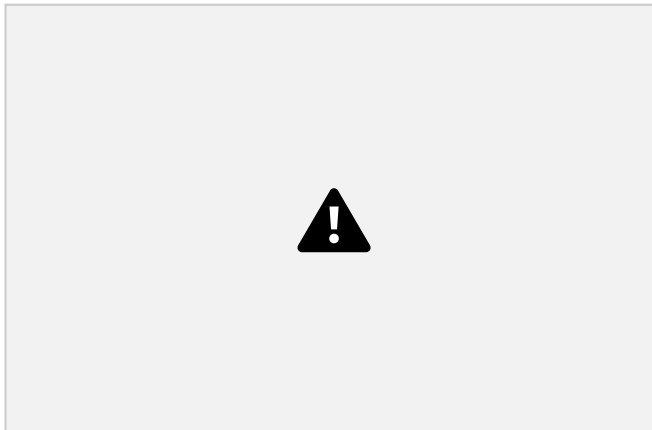
Por todo lo mencionado, podemos decir que en el caso de MySQL se puede hacer uso de la herramienta web que trae incorporada con el ánimo de crear los usuarios.



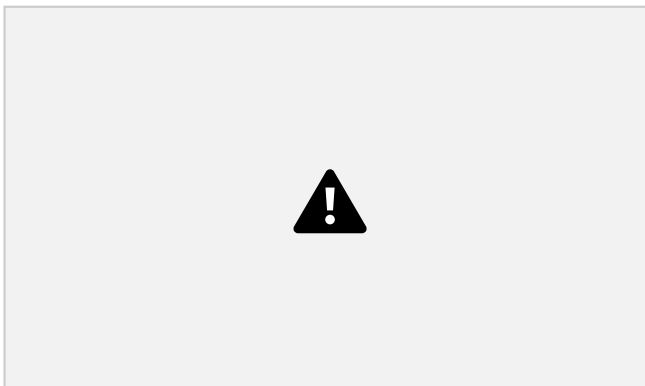
69

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

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:

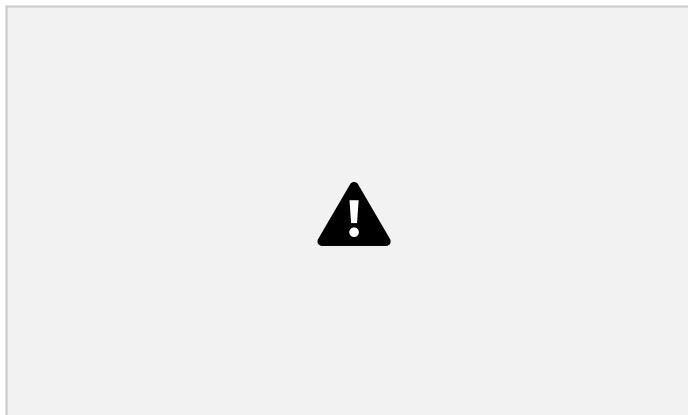


Allí podemos apreciar una interfaz similar a la siguiente

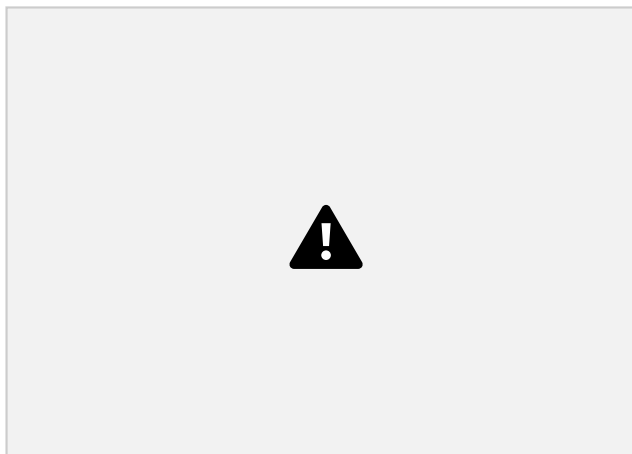


en la cual se digita el nombre del usuario, el servidor al que va a tener derecho de acceso este usuario y la contraseña de registro.

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
```

71

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

```
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")))
```

72

Manual de PHP y MySQL

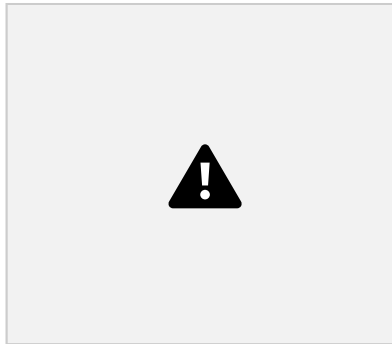
```
{  
    echo "Errores conectando con el servidor";  
    exit();  
}  
else  
{  
    echo "Conexion correcta."<br>;  
}  
  
if (!mysql_select_db("lucho",$conexion))  
{  
    echo "Error conectando con la base de datos."  
    echo "<br>";  
    exit();  
}  
else{  
    echo "Base de datos correcta";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

```
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);
?>

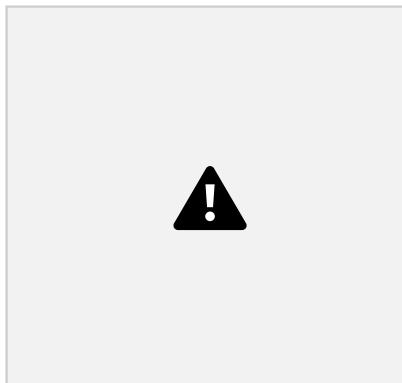
</body>
</html>

```

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

74

Manual de PHP y MySQL



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>";
}

```

75

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

```

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

76

Manual de PHP y MySQL

```
    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);

```

77

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

```

?>
</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"]);
    }
}

```

79

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

```

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 $ssql = "select * from estudiante " . $criterio;
$rs = mysql_query($ssql,$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>";

```

81

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

```

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

```

//construyo la sentencia SQL

```

$ssql = "select * from estudiante " . $criterio . " limit " . $inicio . "," . $TAMA
NO_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 . "
        .
        ;
    }
}

```