



Taller de PHP

Introducción al taller de PHP

En este manual vamos a desarrollar diversas técnicas que se utilizan en PHP para realizar procesos un poco más complejos, los artículos están tratados con un enfoque práctico y son independientes unos de otros. Para leerlos con posibilidad de asimilar los conceptos y técnicas de programación es necesario tener unos conocimientos previos sobre el lenguaje. Por ello, si aun no has aprendido PHP es aconsejable que empieces por el [Manual de programación en PHP](#) antes de dedicar tus esfuerzos a este Taller de PHP.

De todos modos, antes de entrar con los capítulos prácticos vamos a ver una brevísima introducción al lenguaje

Qué es PHP

PHP Es un lenguaje de programación de páginas web del lado del servidor cuyas características principales son la independencia de plataforma y su gratuidad.

Un lenguaje del lado del servidor es aquel que se ejecuta en el servidor web, justo antes de que se envíe la página a través de Internet al cliente. Las páginas que se ejecutan en el servidor pueden realizar accesos a bases de datos, conexiones en red, y otras tareas para crear la página final que verá el cliente. El cliente solamente recibe una página con el código HTML resultante de la ejecución de la PHP.

PHP es uno de los lenguajes que sirven para la programación de scripts del lado del servidor, otros lenguajes muy utilizados son ASP o JSP, que tienen características similares.

Para poder programar en PHP se requiere de un servidor preparado para ello. Como el lenguaje de programación es multiplataforma, cualquiera de los principales servidores web nos servirán para ello. Lo único que tenemos que hacer para preparar nuestro servidor para entender el PHP es descargar de la página de inicio de la tecnología, www.php.net, el módulo específico para el sistema que estamos utilizando. Una vez descargado tendremos que instalarlo en la computadora, siguiendo las instrucciones que podremos encontrar en la misma página. A partir de ese momento nuestro servidor web podrá entender las páginas PHP y nos podremos lanzar a programar en este potente lenguaje.

Las personas que tienen un sistema Windows 98 también pueden instalar un servidor web y prepararlo para PHP. El servidor web más accesible para estos sistemas es el Personal

Web Server, que se encuentra en el mismo CD de windows 98, en el directorio AddOns. Desde la página de PHP se puede encontrar el modulito que permitirá al Personal Web Server, entender las páginas PHP.

Encontrar un proveedor que soporte PHP, para publicar las páginas en Internet es relativamente fácil y barato. PHP está ampliamente difundido entre los servidores Linux o Unix, que no permiten la programación con ASP, lo que hace de PHP el complemento ideal para que sus servidores puedan realizar la programación de páginas con scripts de servidor. Una herramienta muy útil para encontrar el alojamiento PHP más apropiado a nuestras necesidades es nuestro [buscador de alojamiento](#).

Informe de **Miguel Angel Alvarez**
Director DesarrolloWeb.com
Mail: miguel@desarrolloweb.com

Tutorial de sesiones

Si existe una consulta repetida en las listas de PHP, es la relativa al uso de las sesiones. El uso de sesiones es un método ampliamente extendido en cualquier aplicación de cierta entidad. Básicamente una sesión es la secuencia de páginas que un usuario visita en un sitio web. Desde que entra en nuestro sitio, hasta que lo abandona.

El término sesión en PHP, session en inglés, se aplica a esta secuencia de navegación, para ello crearemos un identificador único que asignamos a cada una de estas sesiones de navegación. A este identificador de sesión se le denomina, comunmente, como la sesión.

El proceso en cualquier lenguaje de programación podría ser algo así:

Existe una sesión?

Si existe la retomamos

Si no existe creamos una nueva

Generar un identificador único

Y para que no perdamos el hilo de la navegación del usuario deberemos asociar esta sesión a todas las URLs y acciones de formulario. Podemos tambien crear un cookie que incluya el identificador de sesión, pero es conveniente recordar que la disponibilidad o no de las cookies depende del usuario, y no es conveniente fiarse de lo que un usuario pueda o no tener habilitado.

Lo contado hasta ahora es teoría pura y es aplicable a cualquier lenguaje de programación C, Perl, etc. Los que programamos en PHP4 tenemos la suerte de que toda la gestión de sesiones la hace el mismo PHP. Por lo tanto lo comentado a partir de aquí es solo aplicable a PHP4. Si aún desarrollas PHP3, tendras que crear tus propias librerías de gestión de sesiones o recurrir a alguna de las existentes, como la de <http://phplib.netuse.de/>>PHPLIB.

Para utilizar sesiones en PHP lo primero es inicializarlas. Podemos hacerlo explícitamente, mediante la función session_start(), o al registrar una variable en una sesión mediante session_register('miVariable'). En ambos casos se crea una nueva

sesión, si no existe, o se retoma la sesión actual. Veamos un sencillo ejemplo:

```
<?php
session_start();
echo 'he inicializado la sesión';
?>
```

Esta es la forma más básica, si el usuario tiene los cookies activados, PHP habrá insertado de forma automática la sesión y ésta será pasada de una página a otra sin hacer nada más. Desde un punto de vista práctico la sesión es operativa, pero no vemos nada. Podemos obtener la sesión en cualquier momento mediante la función `session_id()`. Inserta en las sucesivas páginas la siguiente línea para ver si la sesión está disponible:

```
<?php
session_start();
echo 'La sesión actual es: '.session_id();
?>
```

En este caso `session_start()` comprueba en los cookies que existe una sesión y continua con ella, `session_id()` devuelve el identificador actual. Veamos otro ejemplo que, tal vez, te lo aclare un poco más:

```
<?php
session_register('contador');
echo '<a href="'.PHP_SELF.'?'.SID.'">Contador vale: '.++$contador.'</a>';
?>
```

Como dije anteriormente la sesión se crea o recoge mediante `session_start()`, o también cuando se registra una variable de sesión mediante `session_register()`. Si no has utilizado nunca las sesiones, el concepto de variable de sesión, puede resultar un poco abstracto. Básicamente es una variable, como cualquiera de las que gestiona PHP4, pero que reside en un espacio específico en el servidor, junto con el identificador de sesión, y que pertenece únicamente a un usuario. En nuestro ejemplo anterior, registramos la variable `$contador` en la primera línea del script. En la segunda línea, entre otras cosas, cada vez que recargemos la página o hagamos click sobre el enlace, el valor de `$contador` se incrementará en 1.

En esta línea hacemos uso de la variable reservada `$PHP_SELF`, que hace referencia al propio script en ejecución y una constante propia de PHP4, `SID`, que contiene el nombre de la sesión y el identificador de la misma. Si situas el ratón sobre el enlace verás algo parecido a esto en la barra de status del navegador, en la parte inferior:

<http://www.intranet.xxx/session.php?>

Desactiva los cookies en tu buscador y haz la prueba de nuevo, el enlace debería ser algo así:

<http://www.intranet.xxx/session.php?PHPSESSID=86c7a1fefddb6a127005131fc6d3b>

A continuación de la interrogación encontramos la variable `$PHPSESSID`, que es el

nombre por defecto que asigna PHP4 al identificador de sesión y el propio identificador. Por lo tanto \$PHPSESSID, contiene el identificador de sesión. Ya está a tu disposición para cualquier tarea que tengas que realizar.

Podemos averiguar también el nombre de la sesión, o modificarlo, mediante la función session_name(). Veamos una prueba práctica:

```
<?php
session_name('misesion');
session_register('contador');
echo '<a href="'. $PHP_SELF . '?' . SID . '">Contador vale: '. ++$contador.'</a><br>';
echo 'Ahora el nombre es '.session_name().' y la sesión '.$misesion.'<br>';
?>
```

La asignación del nombre de sesión debe realizarse antes que ninguna otra función con sesiones, antes que session_start() o session_register().

Uno de los errores más comunes cuando se utilizan sesiones es dejar líneas en blanco antes de la inicialización de PHP o enviar alguna salida a la pantalla. Para probarlo crea una línea en blanco o con cualquier cosa antes de <?php.

Si tienes los cookies activados, te encontrarás un error de este tipo:

```
Warning: Cannot send session cookie -
headers already sent by (output started at /home/session.php: 2)
in /home/session.php on line 4
```

PHP está informando de que no puede activar los cookies en el navegador del usuario, porque las cabeceras ya han sido enviadas. Simplemente por la existencia de una línea en blanco. Como medida práctica, no dejes espacios ni antes del inicio del script, ni después de la finalización. Te ahorrará muchos disgustos.

Si después de todo lo comentado aún no entiendes para que sirven las sesiones, veamos un ejemplo práctico. Imagina que quisieras crear un sistema de cesta de la compra, en su forma básica podría ser algo así:

```
<?php
session_start();
session_register('itemsEnCesta');
if ($item){
if (!isset($itemsEnCesta)){
$itemsEnCesta[$item]=$cantidad;
}else{
foreach($itemsEnCesta as $k => $v){
if ($item==$k){
$itemsEnCesta[$k] += $cantidad;
$encontrado=1;
}
}
}
if (!$encontrado) $itemsEnCesta[$item]=$cantidad;
}
}
?>
```

```

<html>
<body>
<tt>
<form action="<?=$PHP_SELF."?".SID?>" method="post">
Dime el producto <input type="text" name="item" size="20"><br>
Cuántas unidades <input type="text" name="cantidad" size="20"><br>
<input type="submit" value="Añadir a la cesta"><br>
</form>
<?
if (isset($itemsEnCesta)){
echo'El contenido de la cesta de la compra es: <br>';
foreach($itemsEnCesta as $k => $v){
echo 'Artículo: '.$k.' ud: '.$v.'<br>';
}
}
?>
</tt>
</body>
</html>

```

Una breve explicación. En la línea 4 comprobamos si el usuario ha pasado algún artículo, desde el formulario. En la 5 si el array itemsEnCesta no existe, lo creamos con el nuevo producto y la cantidad indicada. Si el array existe recorreremos su contenido, entre las líneas 8 y 13, y si encontramos un artículo igual, añadimos la cantidad en la línea 10. Si no lo encontramos, es un nuevo artículo, por lo tanto, añadimos el nuevo producto con la correspondiente cantidad a itemsEnCesta en la línea 14.

Y a continuación imprimos el formulario y los resultados, si los hubiera, a partir de la línea 18, donde empieza el HTML.

¿Te imaginas las posibilidades de un sistema de almacenamiento de información de estas características?. No necesitas ficheros, ni bases de datos, ni tienes que andar pasando valores de una página a otra. PHP va gestionando estos datos por nosotros, hasta el momento en que decidamos almacenar la información donde más nos interese.

Estas son las funcionalidades básicas de las sesiones, espero que te halla resultado ilustrativo y no olvides consultar el resto de funciones asociadas al uso de sesiones en el <http://www.php.net/manual/es/ref.session.php> manual de PHP.

En una segunda entrega trataré del almacenamiento de sesiones en base de datos. Hasta pronto.

Informe de **José Valle**
Mail: orion@sarenet.es

Acceso a base de datos con PHPLIB

El acceso a base de datos es una de las tareas más comunes, en cualquier aplicación sobre Internet. Y una de las primeras que cualquier programador trata de automatizar. Y como casi todo en esta vida se les ha ocurrido antes a algunos miles, y en algunos casos lo han resuelto francamente bien.

Como muestra, las clases de base de datos que incluye <<http://phplib.netuse.de/>>PHPLIB entre otras, como las que incorpora para la gestión de sesiones, autenticación, etc.

Si vienes de la programación estructurada, probablemente las clases y la programación orientada a objetos, OOP, te resulten un tanto intimidatorias. Pero francamente el uso de las clases aporta tantas ventajas que merece la pena perder un poco de tiempo en comprender su funcionamiento.

Procuraré no hacer uso de la terminología que se utiliza en la OOP, aunque algunas veces será inevitable. En estos casos pasaré de puntillas sobre estos temas, ya que no es finalidad de este tutorial introducirte en el mundo de los objetos.

Para el uso de DB_Sql, solo tienes que <<http://phplib.netuse.de/download/index.php3>>descargarte la librería de clases y descomprimirlas. No te asustes entre todos esos ficheros, solo nos interesan unos pocos, en concreto los que empiezan por db_. Como podrás ver PHPLIB incluye clases para manejar las bases de datos más populares, entre las que se cuentan las de MySQL, en la que nos centraremos, Oracle y Postgress entre otras.

Para utilizar una clase lo primero que hacemos es crear una instancia de la misma y para ello debemos informar a PHP donde encontrar la clase. Lo haremos de la siguiente forma:

```
include ('/ruta_al_fichero/db_mysql.inc');  
$q= new DB_Sql;
```

Como puedes ver para ello hemos utilizado la palabra new y lo hemos asignado, a una variable. En realidad esto es un objeto, una instancia de la clase DB_Sql.

Ahora inicializaremos los parámetros de conexión a la base de datos, modificando algunas variables definidas dentro de la clase:

```
$q->Host = "tuHost";  
$q->Database = "tuBaseDeDatos";  
$q->User = "tuUsuario";  
$q->Password = "tuPassword";
```

Por supuesto, puedes asignar estos parámetros dentro de la clase a las correspondientes variables, de esta forma los datos de conexión siempre serán los mismos.

Hasta el momento deberías tener algo como esto:

```
<?php  
include ('/ruta_al_fichero/db_mysql.inc');  
$q= new DB_Sql;  
  
$q->Host = "tuHost";  
$q->Database = "tuBaseDeDatos";
```

```
$q->User = "tuUsuario";  
$q->Password = "tuPassword";  
?>
```

Y llego el momento de hacer algo práctico con nuestra clase. Supongamos una base de datos con la siguiente estructura:

```
create table amigos (  
id int(11) default '0' not null auto_increment,  
nombre varchar(25),  
apellidos varchar(25),  
dirección varchar(50),  
ciudad varchar(50),  
pais varchar(50),  
primary key (id)  
);
```

Vamos a realizar un query a la base de datos, usando la clase. La construcción del query se hace de la misma forma que si atacaramos directamente a la base de datos, con las funciones de PHP:

```
$query='select * from amigos';  
$q->query($query);
```

Con esto, si todo fué bien, habremos conseguido los registros que cumplen la condición de la tabla amigos. Pero, ¿como podemos asegurarnos de que hemos obtenido algún registro?. Bien, por un lado deberemos asegurarnos de que la conexión con la base de datos se ha establecido. DB_sql establece una conexión persistente al efectuar la consulta, por lo tanto si query devuelve algún valor, sabremos que la conexión se ha establecido. Para asegurarnos de que la consulta ha sido exitosa, deberemos tener en cuenta que algunas veces una consulta puede ser verdadera, pero no devolver ningún resultado, caso muy común al realizar un select. Es por lo tanto una buena práctica verificar este punto, así es como quedaría nuestro código:

```
$query='select * from amigos';  
if (!$q->query($query)){  
echo 'Lo siento no se pudo establecer la conexión<br>';  
}else{  
if (!$q->num_rows()){  
echo 'Lo siento no se ha obtenido ningún resultado<br>';  
}else{  
echo 'Hey! hay '.$q->num_rows().' registros que cumplen la condición';  
}  
}
```

Esto esta muy bien, veamos ahora como manejar los registros y campos de forma individual. Para ello recorreremos los resultados e iremos mostrando los datos que nos interesen uno a uno:

```

$query='select * from amigos';
if (!$q->query($query)){
echo 'Lo siento no se pudo establecer la conexión<br>';
}else{
if (!$q->num_rows()){
echo 'Lo siento no se ha obtenido ningún resultado<br>';
}else{
echo '<pre>';
echo 'Hey! hay '.$q->num_rows().' registros que cumplen la condición';
while ($q->next_record()!=0){
echo 'El id es: '.$q->f('id').'\n';
echo $q->f('nombre').' '.$q->f('apellidos').'\n';
}
echo '</pre>';
}
}
}

```

De la misma forma que hacemos un select, haremos un insert, update o delete. Obviamente estos tres últimos no devuelven resultados, por lo tanto para saber si nuestros queries han funcionado recurriremos a la función `affected_rows` de la clase `DB_Sql`, de esta forma:

```

$query='delete from amigos where ciudad=\'Medellín\'';
if (!$q->query($query)){
echo 'Lo siento no se pudo establecer la conexión<br>';
}else{
if (!$q->affected_rows()){
echo 'Lo siento no se encontro ningún registro que cumpla la condición<br>';
}else{
echo 'Se han eliminado '.$q->affected_rows().' registros';
}
}
}

```

Como puedes comprobar, el uso de una clase no es nada complicado. Solo debes preocuparte por conocer los métodos, las funciones de la clase que tienes disponibles y lo que hacen. Habitualmente las clases están suficientemente documentadas y te muestran los valores de entrada y salida requeridos. Además es una buena forma de mejorar tu estilo de programación y adquirir nuevos conocimientos.

El uso básico de las clases que incorpora PHPLIB para otras bases de datos es básicamente igual, puede que más o menos funciones, aunque las básicas son iguales en todos los casos. Es importante que revises la documentación, para comprobar las posibles diferencias.

Ten en cuenta que cualquier llamada del tipo `$q->loquesea()`, es una llamada al método de una clase y que algo del tipo `$q->otracosa` es una variable de la clase. Con paréntesis, un método, como una función normal de PHP y sin ellos una variable. Dentro de la clase la llamada a las funciones y variables se hace con la palabra reservada `this`. Por lo tanto cuando te encuentres un `$this->algo`, fíjate si tiene o no paréntesis a continuación, y así sabrás si es una llamada a un método de la clase o alguna operación con variables.

Y esto es todo de momento, espero que te sirva de ayuda. Hasta pronto.

Informe de **José Valle**
Mail: orion@sarenet.es

Gestión de archivos por PHP

El tratamiento de archivos resulta ser una práctica muy común en cualquier sitio web. Muy a menudo nos vemos en la necesidad de procesar un texto para cambiarle el formato, buscar una cadena en su interior o cualquier otro tipo de operación.

PHP propone un sinfín de [funciones para la gestión de archivos](#) que van desde las más elementales de apertura, lectura y cierre a otras más rebuscadas como el cálculo de espacio en el disco duro, tamaño del archivo, gestión de derechos de acceso...

En este artículo pretendemos mostraros cuáles son las funciones más esenciales para el tratamiento de archivos para posteriormente ejemplificarlas en un par de scripts que os pueden resultar útiles:

Funciones de gestión de archivos

Función	Descripción	Sintaxis
copy	Copia un archivo	<code>copy(\$origen,\$destino)</code>
rename	Cambia el nombre del archivo de <i>\$antes</i> a <i>\$despues</i>	<code>rename(\$antes,\$despues)</code>
unlink	Borra el archivo	<code>unlink(\$archivo)</code>

Funciones para la lectura de archivos

Función	Descripción	Sintaxis
fopen	Abre un archivo y le asigna un identificador id. Veremos el modo más adelante	<code>\$id = Fopen(\$archivo, \$modo)</code>
fgets	Lee una línea de un archivo hasta un numero máximo de caracteres	<code>fgets(\$id,\$max)</code>
fwrite	Escribe una cadena dentro del archivo	<code>fwrite(\$id, \$cadena)</code>
fseek	Avanza o retrocede el puntero del archivo un cierto numero de posiciones	<code>fseek(\$id,\$posiciones)</code>
feof	Comprueba si el puntero que lee el archivo ha llegado al final	<code>feof(\$id)</code>
fpassthru	lee completamente el archivo y lo muestra	<code>fpassthru(\$id)</code>
fclose	Cierra el archivo abierto previamente	<code>fclose(\$id)</code>

Las operaciones más elementales, copia, borrado y cambiar el nombre, requieren únicamente el nombre (y path) del archivo sobre el cual se ejerce la operación. Para operaciones más complejas, como la lectura de líneas o la escritura de texto dentro del archivo, se requiere de una previa apertura del archivo al cual le asignaremos un indentificador *\$id*.

Una vez abierto el archivo, podremos desplazarnos a lo largo de él por medio de un puntero imaginario que avanza o retrocede por las líneas de texto y mediante el cual nos

situaremos en el lugar escogido para insertar, modificar o simplemente copiar una cadena.

Existen distintos modos de apertura que nos permiten definir las acciones que podemos realizar sobre el archivo. Aquí os mostramos los diferentes modos que, como veréis, son de lo más variado:

Modos de apertura de archivos

Sintaxis	Descripción
'r'	Sólo lectura
'r+'	Lectura y escritura
'w'	Sólo escritura
'w+'	Lectura y escritura. Suprime el contenido anterior si se escribe. El archivo es creado si no existe.
'a'	Sólo escritura. El archivo es creado si no existe y el puntero se coloca al final.
'a+'	Lectura y escritura. El archivo es creado si no existe y el puntero se coloca al final.

A notar que si tratamos con archivos en binario hemos de colocar una *b* delante del modo (ej. *ba*, *bw+*, ...)

Recordamos que esta lista no es más que una recopilación y que muchas [otras funciones relacionadas](#) pueden sernos también útiles.

Informe de **Rubén Alvarez**

Mail: ruben@desarrolloweb.com

Verificar la existencia de una URL

Ya hemos visto las [funciones de gestión de archivos](#) más comúnmente utilizadas en PHP. Sirvámonos de la más clásica de todas ellas, *fopen*, para crear un script que verifique la existencia de una URL.

Este tipo de script puede ser utilizado para múltiples propósitos: Detectar si los enlaces están rotos, verificar una etapa de inscripción en un formulario...

En este caso, hemos simplificado al máximo su contenido de manera a poner evidencia su funcionamiento. Otro tipo de mejoras tales como la verificación de la extensión del archivo (.asp, .php, ...) o del protocolo de transferencia (http, ftp...) pueden ser introducidas afín de personalizar su uso para distintas aplicaciones.

A continuación podéis ver como quedaría el script:

```

<?
function verificar_url($url)
{
    //abrimos el archivo en lectura
    $id = @fopen($url,"r");
    //hacemos las comprobaciones
    if ($id) $abierto = true;
    else $abierto = false;
    //devolvemos el valor
    return $abierto;
    //cerramos el archivo
    fclose($id);
}
?>
<html>
<head>
    <title>Verificacion de URL</title>
</head>
<body>
<?
if (!isset($url))
{
?>
    <form action="enlace.php" method="post">
    Indica tu URL: <br>
    <input type="Text" size="25" maxlength="100" name="url" value="http://">
    <input type="Submit" value="Verificar!"
    </form>
<?
}
else
{
    $abierto = verificar_url($url);
    if ($abierto) echo "La URL existe!";
    else echo "La URL no existe o es inaccesible...";
}
?>
</body>
</html>

```

Ejecutar ejemplo

Hemos introducido en el mismo script que se encarga de verificar la URL el formulario que se encarga de recogerla. Así, podemos dividir el script en dos partes: Una primera que se encarga de recoger la URL en un campo texto y una segunda que es la que verdaderamente evalúa la existencia de la URL.

Para hacer más aplicable el script hemos dejado la evaluación propiamente dicha en forma de función que podréis copiar y pegar en vuestra librería particular. Como podéis ver, el modo de operar es extremadamente sencillo:

- Abrimos el archivo remoto por medio de la función *fopen* en modo solo lectura. A notar que precediendo a la función *fopen* hemos introducido un símbolo arroba @ cuyo cometido es el de ignorar el posible error producido por la sentencia. Esto nos evita ver el mensaje de error que es mostrado cuando la URL no existe.
- Verificamos que el identificador de apertura *\$id* no esta vacío. Es en este punto donde podemos implementar a nuestra función las mejoras de las que hemos hablado, las cuales le confieren la verdadera utilidad.
- Devolvemos un valor *true* o *false* dependiendo del éxito de la conexión.

Como podéis ver el script no reviste ninguna dificultad y puede sernos muy práctico.

Informe de **Rubén Alvarez**

Mail: ruben@desarrolloweb.com

Lectura secuencial de archivos con PHP

Sigamos con nuestro aprendizaje práctico del [uso de ficheros en PHP](#). Ya hemos visto cómo abrir un archivo por medio de la función *fopen* con un ejemplo práctico de [cómo verificar una URL](#). El paso siguiente es aprender a leer el contenido del archivo, tarea que llevaremos a cabo por medio de la función *fgets*.

Esta función se encarga de leer línea a línea el contenido de un archivo texto por lo que su utilización ha de ser incluida dentro de una estructura de tipo bucle.

En el ejemplo que os mostramos a continuación nos hemos servido de esta lectura secuencial para localizar dentro del texto una cadena cualquiera a la que, a continuación, le cambiamos el formato para ponerla en negrita por medio de la etiqueta . Esto nos puede resultar útil si llevamos a cabo búsquedas internas en nuestro sitio y queremos resaltar la cadena de búsqueda en el texto de la página encontrada.

Evidentemente, la utilidad de *fgets* resulta ser mucho más amplia. Podemos emplearla, por ejemplo, con archivos remotos para extraer las etiquetas meta o para muchas otras cosas que se nos puedan ocurrir.

Aquí os proponemos el script:

```

<?
function negrita($path,$cadena)
{
    //Iniciamos la variable
    $texto = "";
    //Abrimos el archivo en modo lectura
    $fp = fopen($path,"r");
    //Leemos linea por linea el contenido del archivo
    while ($linea= fgets($fp,1024))
    {
        //Sustituimos las ocurrencias de la cadena que buscamos
        $linea = str_replace($cadena,"<b>$cadena</b>", $linea);
        //Anadimos la linea modificada al texto
        $texto .= $linea;
    }
    return $texto;
}
//Definimos el path y la cadena
$path="escribe el camino de acceso a tu archivo";
$cadena = "escribe tu cadena";
//Llamamos la funcion
$texto = negrita ($path,$cadena);
//Mostramos el texto
echo $texto;
?>

```

Podéis ver el resultado de esta función en una variante del script donde hemos incluido un formulario para recibir el parámetro *cadena* y que busca las ocurrencias dentro del texto de este mismo artículo:

Introduce la cadena de búsqueda:

El script es utilizado en forma de función para facilitaros su empleo y almacenamiento. Su modo de actuar es el siguiente:

- Inicializa la variable *texto* en la cual iremos almacenando las líneas leídas en el bucle.
- Abre el archivo (local o remoto) en modo lectura por medio de la función *fopen*.
- Lee una por una las líneas del archivo hasta una longitud de 1024 caracteres y reemplaza las posibles ocurrencias de la cadena de búsqueda por la misma cadena colocada entre las etiquetas ** y ** por medio de la función *str_replace*. El texto, modificado o no, es almacenado en la variable *texto*.
- Devolvemos la variable *texto* como resultado de la función.

El resto de script es simplemente un ejemplo de llamada a la función donde los parámetros *path* y *cadena* han de ser especificados.

Escritura en archivos con PHP

Siguiendo con la gestión de archivos por medio de PHP, en este artículo veremos los pasos elementales para la creación y escritura de un archivo texto por medio de esta tecnología de lado servidor. Tras haber visto como funciona la [lectura secuencial de un archivo](#), podemos imaginar que escribir sobre éste no debe de resultar mucho más complicado.

Por otra parte, las posibilidades que estas dos operaciones nos pueden ofrecer conjuntamente son realmente sorprendentes. Sin ir más lejos, y guardando las distancias, escribir y leer archivos puede en cierta manera sustituir muy primariamente a una base de datos. En efecto, si por diversas razones (hosting, presupuesto, conocimientos...) nos resulta difícil la puesta en marcha de una base de datos elemental, siempre podremos solventar el inconveniente almacenando nuestros datos en archivos que luego podrán ser leídos. Por supuesto, este método no tiene nada de seguro ni de versátil y sólo es valido para un sitio sin información confidencial y con poca cantidad de datos.

Podemos pensar también en crear documentos dinámicos a partir de datos introducidos en un formulario: cartas, páginas HTML y otros.

Otro ejemplo particularmente práctico es la creación dinámica de archivos que nos ahorren recursos de servidor. Imaginemos que tenemos una página, o archivo, en nuestro sitio que carga muy frecuentemente y que realiza constantemente llamadas a bases de datos o ejecuta scripts medianamente largos. Si el contenido que estamos mostrando es el mismo para todos los usuarios y no tiene necesidad de ser actualizado constantemente, podemos contentarnos con crear un script accesorio que ejecute una única vez el script principal y que almacene su resultado en forma de archivo HTML que será en realidad el que mostraremos a nuestros visitantes. De esta forma, evitamos por una parte la ejecución masiva de un mismo script con el consiguiente ahorro de recursos y por otra automatizamos la actualización de una determinada página o sección ejecutando periódicamente el script accesorio.

La escritura de archivos pasa, como es de esperar, por la previa [apertura de archivo en un modo apropiado](#). Una vez abierto, el paso siguiente será introducir por medio de la función *fwrite*, o su alias *fputs*, la cadena que deseamos incluir en nuestro archivo.

Para ejemplificar esta nueva función de escritura y combinarla con la de lectura, *fgets*, os proponemos este contador inspirado en una nota de la [página oficial de PHP](#):

```
<?
function incremento_contador($archivo)
{
// $archivo contiene el numero que actualizamos
$contador = 0;

//Abrimos el archivo y leemos su contenido
$fp = fopen($archivo,"r");
$contador = fgets($fp, 26);
fclose($fp);

//Incrementamos el contador
++$contador;

//Actualizamos el archivo con el nuevo valor
$fp = fopen($archivo,"w+");
fwrite($fp, $contador, 26);
fclose($fp);

echo "Este script ha sido ejecutado $contador
veces";
}
?>
```

[Aquí](#) podéis ver el resultado producido cuando llamamos a esta función.

Como en otros ejemplos, el script es expresado en forma de función para que sea más sencilla su reutilización. Las etapas que llevamos a cabo son verdaderamente cortas y comprensibles:

- Iniciamos nuestra variable *contador*.
- Abrimos el archivo en modo lectura y extraemos el valor actual del contador leyendo la primera y única línea. Cerramos el archivo.
- Aumentamos de una unidad el valor de *contador*.
- Abrimos el archivo y lo sobrescribimos (modo +w) con el valor contador modificado.

Ni que decir tiene que para que este tipo de scripts funcionen, el archivo al que queremos acceder ha de estar autorizado para lectura y escritura.

La función *fwrite* puede ser utilizada también para enviar datos en otros tipos de aperturas como son las de sockets o de programas. Pero esto ya es otra historia...

Informe de **Rubén Alvarez**
Mail: ruben@desarrolloweb.com

Usuarios activos con PHP

En nuestro manual de PHP abordamos en su momento el [uso de sesiones](#) y dimos algún

ejemplo práctico en el que este tipo de variables pueden ser utilizadas para dar a nuestro sitio un aspecto más dinámico.

Muchos de vosotros habéis podido ver en ciertos sitios un contador de usuarios que se encuentran en ese momento navegando por las mismas páginas que vosotros. Si os habéis fijado bien, habréis podido observar que, para la mayoría de los casos (sino la totalidad), el sitio en cuestión está programado con ASP como lenguaje de servidor.

Efectivamente, ASP permite una gestión más accesible de las sesiones por medio del famoso archivo [global.asa](#). Esto no quiere decir sin embargo, que PHP es incapaz de realizar el mismo tipo de tareas sino que, más bien, hemos de trabajar un poco más para conseguirlas. En efecto, todos los lenguajes tienen sus pros y sus contras y, en este caso particular, ASP resulta más versátil aunque hay que admitir que, con su versión 4, PHP ha mejorado mucho en todo lo que respecta al tratamiento de sesiones.

Aquí os mostramos una forma sencilla de contabilizar los usuarios activos de vuestro sitio usando para ese propósito una tabla. Dentro de dicha tabla, iremos almacenando los distintos números IP de los visitantes de nuestro sitio y la hora y fecha en la que el visitante ha ejecutado por última vez el script.

Asimismo, la tabla ha de ir borrando progresivamente las sesiones que no hayan sido renovadas en un tiempo que nosotros consideremos límite. Nosotros hemos fijado dicho límite en 24 minutos que es el tiempo máximo tomado por defecto por PHP para suprimir los datos de una sesión. Para modificar este tiempo de vida máxima de una sesión puede hacerse en el *php.ini* a partir del parámetro *session.gc_maxlifetime* donde expresaremos dicho plazo en segundos. Ojo, este tiempo máximo es restaurado a su valor inicial cada vez que el usuario realiza una petición al servidor, esto quiere decir que un visitante podrá navegar cuanto tiempo quiera por el sitio guardando la misma sesión siempre y cuando no se quede más de 24 minutos sin realizar ningún tipo de acción.

Para el correcto funcionamiento del script, es necesario antes de nada crear una tabla en nuestra base de datos. Esta sentencia SQL puede ayudaros en la tarea:

```
CREATE TABLE control_ip (  
ip VARCHAR(15) NOT NULL,  
fecha INT(14) UNSIGNED NOT NULL,  
INDEX (ip)  
);
```

Como veis, el campo ip, que almacena el número IP del visitante, está indexado. Esto nos permitirá una selección rápida. En contrapartida, como todo campo indexado, su tamaño en memoria será doblado lo cual no tiene mucha importancia ya que la tabla tendrá un número de registros bastante limitado. Lo importante en efecto es que el script se ejecute rápidamente sin consumir demasiados recursos del servidor, sobretodo teniendo en cuenta que se trata de un código que será sistemáticamente ejecutado en cada una de las páginas del sitio.

Pasemos a continuación a mostrar el script que utilizaremos:

<?

```
////////////////////////////////////  
//USUARIOS ACTIVOS  
//Calcula el numero de usuarios activos  
////////////////////////////////////
```

```
function usuarios_activos()
```

```
{
```

```
    //permitimos el uso de la variable portadora del numero ip en nuestra funcion  
    global $REMOTE_ADDR;
```

```
    //asignamos un nombre memotecnico a la variable
```

```
    $ip = $REMOTE_ADDR;
```

```
    //definimos el momento actual
```

```
    $ahora = time();
```

```
    //conectamos a la base de datos
```

```
    //Usad vuestros propios parametros!!
```

```
    $conn = mysql_connect($host,$user,$password);
```

```
    mysql_select_db($db,$conn);
```

```
    //actualizamos la tabla
```

```
    //borrando los registros de las ip inactivas (24 minutos)
```

```
    $limite = $ahora-24*60;
```

```
    $ssql = "delete from control_ip where fecha < ".$limite;
```

```
    mysql_query($ssql);
```

```
    //miramos si el ip del visitante existe en nuestra tabla
```

```
    $ssql = "select ip, fecha from control_ip where ip = '$ip'";
```

```
    $result = mysql_query($ssql);
```

```
    //si existe actualizamos el campo fecha
```

```
    if (mysql_num_rows($result) != 0) $ssql = "update control_ip set fecha = ".$ahora."  
where ip = '$ip'";
```

```
    //si no existe insertamos el registro correspondiente a la nueva sesion
```

```
    else $ssql = "insert into control_ip (ip, fecha) values ('$ip', $ahora)";
```

```
    //ejecutamos la sentencia sql
```

```
    mysql_query($ssql);
```

```
    //calculamos el numero de sesiones
```

```
    $ssql = "select ip from control_ip";
```

```
    $result = mysql_query($ssql);
```

```
    $usuarios = mysql_num_rows($result);
```

```
    //liberamos memoria
```

```
    mysql_free_result($result);
```

```
    //devolvemos el resultado
```

```
    return $usuarios;
```

```
}
```

?>

Podéis observar, como viene siendo norma, que el script es expresado en forma de función. Después de definir la IP y el momento en el que el script está siendo ejecutado, pasamos a interaccionar con la base de datos. Dentro de este proceso, podemos distinguir distintas fases:

- Conexión con la base de datos
- Barrido de la tabla para eliminar los registros obsoletos.
- Inserción o actualización de un registro dependiendo de si el visitante es nuevo o no.
- Cómputo del número de registros de la tabla

La función finaliza liberando el espacio de memoria utilizado en sus consultas y enviando el resultado del cálculo efectuado.

En líneas generales el script es de fácil comprensión. Puede que alguna de las funciones empleadas os sea desconocida. Si es así, acudid al la [pagina oficial de PHP](#) donde podréis encontrar detalles en cuanto a su empleo.

Para sacar el valor proporcionado por la función a nuestro script principal tendremos que realizar una llamada clásica del tipo:

```
$active_users = usuarios_activos();
```

He aquí en definitiva un script sencillo que puede dar a vuestro sitio una imagen un poco más dinámica. Además, podéis utilizarlo y mejorarlo para crear vuestro propio sistema de estadísticas internas. Eso os lo dejamos a vosotros...

Informe de **Rubén Alvarez**
Mail: ruben@desarrolloweb.com

Taller de paso de variables por URL

Este taller resultará un poco básico para algunas personas, ya que no vamos a explicar nada exclusivamente nuevo y que no hayamos comentado ya en otros manuales. Sin embargo, seguro que muchas otras personas agradecerán esta explicación para contestarse algunas dudas a la hora de hacer páginas avanzadas con ASP o PHP.

Resulta que me han llegado algunas duda a mi correo de personas que no comprendían muy bien la manera de realizar páginas que, dependiendo de la variable que reciban por la URL presenten una información u otra. A continuación podemos ver estas dudas para aclararnos.

Duda 1: (Sobre el lenguaje ASP, aunque da igual, ya que la metodología de uno y otro lenguaje son las mismas. Sólo difiere la sintaxis.)

Hola me llamo Orel y necesito si me pueden dar una ayudita con mi pagina web. Bueno primero que nada estoy creando mi paina en asp y no entiendo cómo poner un enlace por ejemplo:

Cuando colocan varios enlaces y todos llevan a la misma pagina ver.asp lo que cambia es ?Id=emuladores o ?Id=Roms etc.

No se cómo hacer eso.

Por que si yo pongo la pagina ver.asp?Id=emuladores Me aparece la pagina de emuladores mientras que si pongo ver.asp solo me muestra la pagina de inicio ¿Como hacen eso de que con la misma pagina puedan crear varias?

Duda 2: (Esta duda es de PHP, pero da igual el lenguaje para el que estemos trabajando, en el fondo es la misma duda)

Como se que vosotros sabéis mucho de php, os quería comentar una duda que tengo. ¿Que necesito para crear una web de carátulas en la cual las carátulas salgan en una página que sea una especie de "plantilla", en la cual solo cambie la carátula? Éste tipo de sistemas los he visto en webs de carátulas y son de la forma:

<http://nombredelaweb/caratula.php?id=587>

<http://nombredelaweb/caratula.php?id=351>

También cambia el texto y otras cosas.

Respuestas

Pues bien, planteadas las dudas, primero decir que en DesarrolloWeb tenemos todos los contenidos necesarios para aprender técnicamente a utilizar estas variables que se pasan por la URL. Es importante que conozcamos la sintaxis para poder hacer los ejercicios donde dudamos.

Aquí tenemos los enlaces:

- Para [aprender a pasar y recoger variables por la URL en ASP](#)
- [Lo mismo pero en PHP](#).

Una vez aprendida la parte técnica podemos pensar en cómo hacer estos sistemas en ambos lenguajes. Lo primero que debemos saber es que esas variables que recibimos por parámetro las podemos utilizar para cualquier cosa. En la duda 1 las vamos a utilizar para mostrar una información de una sección y en la duda 2 las vamos a utilizar para extraer la información de la carátula que se desea ver. Creo que lo mejor sería ver cada uno de los dos casos, en el lenguaje que se preguntan.

Respuesta de la Duda 1, para ASP

Necesitamos extraer el valor de la variable que nos pasan por parámetro y hacer unas cosas u otras en función de dicho valor. Veamos a grandes rasgos el código que utilizaría.

```
<%
```

```
'extraigo el valor de la sección que se desea ver
seccion_ver = request.QueryString("id")
```

```
'Dependiendo de la sección muestro unos u otros contenidos
```

```
if seccion_ver = "emuladores" then
```

```
'entonces coloco todo lo que queramos mostrar en la sección emuladores
```

```
%>
```

```
<h1>Sección de emuladores</h1>
```

```
<p>Bienvenidos a mi sección de emuladores, donde puedes conocer todos los principales
```

```

emuladores del mercado y descargarte los programas.
<p>... (En fin, todo lo que queramos escribir en esta sección)
<%
elseif seccion_ver = "Roms" then
'entonces coloco todo lo que queramos en la sección de Roms
%>
<h1>Sección de Roms</h1>
<p>Bienvenidos a mi sección de Roms, tenemos más de 500 roms de juegos de todos los
tiempos. Cada uno con enlace para descarga e instrucciones de uso.
<p>... (En fin, todo lo que queramos escribir en esta sección)
<%
else
'paso por aquí si no era la sección emuladores ni Roms
'entonces hago lo que desee, que en la duda era simplemente mostrar la página principal.
Response.redirect("index.asp")
'esta instrucción te redirecciona a la página principal de tu sitio web, si es que se llamaba
index.asp y está en el mismo directorio que la página actual.
end if
%>

```

Como se puede comprobar, dependiendo de lo que se reciba por parámetro se mostrará una sección u otra. Si no se recibe nada por parámetro o el valor no corresponde a ninguna de las secciones que hemos definido, se redirecciona a la página `index.asp`, que suponemos que es la página principal del sitio web.

Respuesta de la Duda 2, para PHP

Este ejemplo es, si cabe, más simple que el primero. La forma de trabajar típica que se realiza con un ejemplo como este es que todas las carátulas estén en una tabla de una base de datos. Entonces, cada vez que se accede a la página `caratula.php` se recibe el identificador de la carátula que se quiere visualizar en una variable en la URL, de esta manera: `caratula.php?id=12`. Así estamos accediendo la página que muestra carátulas y le decimos que la carátula que deseamos ver es la que tiene el identificador (id) igual a 12.

Supongamos que metemos la información de la carátula en una tabla que tiene esta forma:

Tabla caratula

id	Identificador de la carátula
titulo	Título del disco
imagen	Nombre del archivo imagen de la carátula
info	Descripción complementaria de la carátula

El código en PHP para hacer algo así sería el siguiente. En este código vamos a obviar algunas instrucciones de conexión con la base de datos que realmente no vienen al caso. Recordamos que en el [Manual de PHP I](#) tenemos todas las explicaciones para aprender a manejar bases de datos.

```

<?
//Extraigo el identificador de la carátula a mostrar
$id_caratula = $_HTTP_GET_VARS["id"];

//creo la sentencia SQL que extrae datos de esa carátula
$sql = "select * from caratula where id=" . $id_caratula;

//obtengo los datos de la carátula
$rs = mysql_query($sql);
$fila = mysql_fetch_object($rs);

//muestro los datos de esa carátula
?>
Disco: <?echo $fila->titulo;?>
<br>
Imagen: 
<br>
Descripción: <?echo $fila->info;?>

<?
//cierro la base de datos
mysql_free_result($rs);
//... todo lo que haga falta para terminar la página...
?>

```

Conclusión

Con esto acabamos. Esperamos que os sirva de ayuda este artículo o, por lo menos, haya resuelto vuestras dudas sobre la utilización de variables en la URL tanto en ASP como en PHP. Sólo pedir disculpas porque los ejemplos que he utilizado no son páginas completas al 100%, ya que faltan las cabeceras de las páginas y otras instrucciones que no venían al caso. Además, como son ejemplos parciales, se me habría podido escapar alguna falta de sintaxis, al no poderse poner en marcha los scripts así, de manera parcial.

Lo importante es que hayáis cogido la idea y entendido mejor para qué puede servir pasar variables en la URL y cómo hacen los desarrolladores para que la página sea distinta dependiendo de los valores de las variables. Espero que haya sido así.

Informe de **Miguel Angel Alvarez**
 Director DesarrolloWeb.com
 Mail: miguel@desarrolloweb.com

Programas de libre distribución en PHP

El lenguaje de programación de páginas dinámicas del servidor multiplataforma y totalmente gratuito, [PHP](#), tiene varias ventajas que no nos hemos cansado de señalar constantemente. Entre estas ventajas hay una que ahora mismo nos interesa comentar. Se trata de que existen en Internet multitud de programas gratuitos creados en PHP para realizar una buena cantidad de tareas habituales. Estos programas los podemos instalar en nuestro sistema libremente, por ser de libre distribución, y disfrutar de sus funcionalidades en nuestros servidores, insistimos, gratuitamente.

Hay muchos programas de libre distribución en PHP. Ejemplos típicos son servidores de banners, servidores de sistemas de afiliación, servidores orientados al control de bases de datos, sistemas de agenda, sistemas de chat, comercios electrónicos... y un largo etcétera que no vamos a alargar en estos momentos. Cualquier persona que desee informarse de la lista completa de scripts disponibles en PHP puede acercarse a la dirección http://www.hotscripts.com/PHP/Scripts_and_Programs/ que corresponde con la página de programas PHP que ofrece el sitio [Hotscripts.com](http://www.hotscripts.com/). En Hotscripts nos presentan multitud de programas, unos gratuitos y otros de pago, pero en el caso de PHP la mayoría son gratuitos, al contrario de lo que ocurre en ASP, donde la mayoría tienen un precio, en ocasiones excesivamente elevado.

Hay un lugar muy típico de Internet donde fomentan y ayudan al desarrollo de programas de libre distribución es [SourceForge](http://sourceforge.net/), con dirección <http://sourceforge.net/>. Se trata de un servicio gratuito para desarrolladores de código abierto que ofrece fácil acceso a herramientas para trabajo en grupo, tableros de anuncios, foros, listas de discusión, seguimiento de errores, archivado de versiones, libre alojamiento de páginas web... como podemos ver todo lo que necesite un desarrollador para la publicación a todo lujo de sus aplicaciones gratuitas. En [SourceForge](http://sourceforge.net/) "viven" muchos proyectos interesantes, que merecería la pena explicar por separado.

[SourceForge](http://sourceforge.net/) es, por tanto, un lugar ideal para encontrar programas para PHP, ya que la mayoría de desarrollos en PHP son de libre distribución. Veamos rápidamente algún ejemplo:

[phpMyAdmin](#)

Es un gestor de base de datos MySQL, a través del cual podemos administrar todos los procesos de nuestra base de datos. Muy útil para tenerlo instalado en nuestro alojamiento PHP, ya que, la base de datos MySQL se tiene que administrar por consola y por línea de comandos y a veces el proveedor no nos ofrece esa posibilidad de conexión. Además, porque nos hace más fácil administrar los datos que con sentencias SQL, como se hace por línea de comandos.

Podemos [aprender más sobre phpMyAdmin y su instalación](#) en un artículo de DesarrolloWeb.com

[phpAdsNew](#)

Un servidor de banners que tiene las funcionalidades de los mejores productos comerciales. Posibilidad de crear clientes, campañas, banners, y administrar las impresiones de banners de nuestro sitio de la manera que mejor queramos. Mantiene unas excelentes estadísticas de impresiones y clicks, hace reports, etc.

[PHP-Nuke](#)

Un sistema completo para la administración de un portal web. Con este sistema podemos administrar un buscador de enlaces, noticias y artículos en general, usuarios registrados, encuestas, etc. Se trata de un software muy popular, ya que existen muchos desarrolladores que han adoptado esta solución para crear su propio portal sin necesidad de muchos esfuerzos en el tiempo de desarrollo.

La lista es interminable y los programas que se ofrecen son realmente interesantes. Sin duda en PHP hay muchos desarrolladores generosos que han trabajado duro para presentarnos estas joyas gratuitas, con las que podemos olvidarnos del desarrollo de

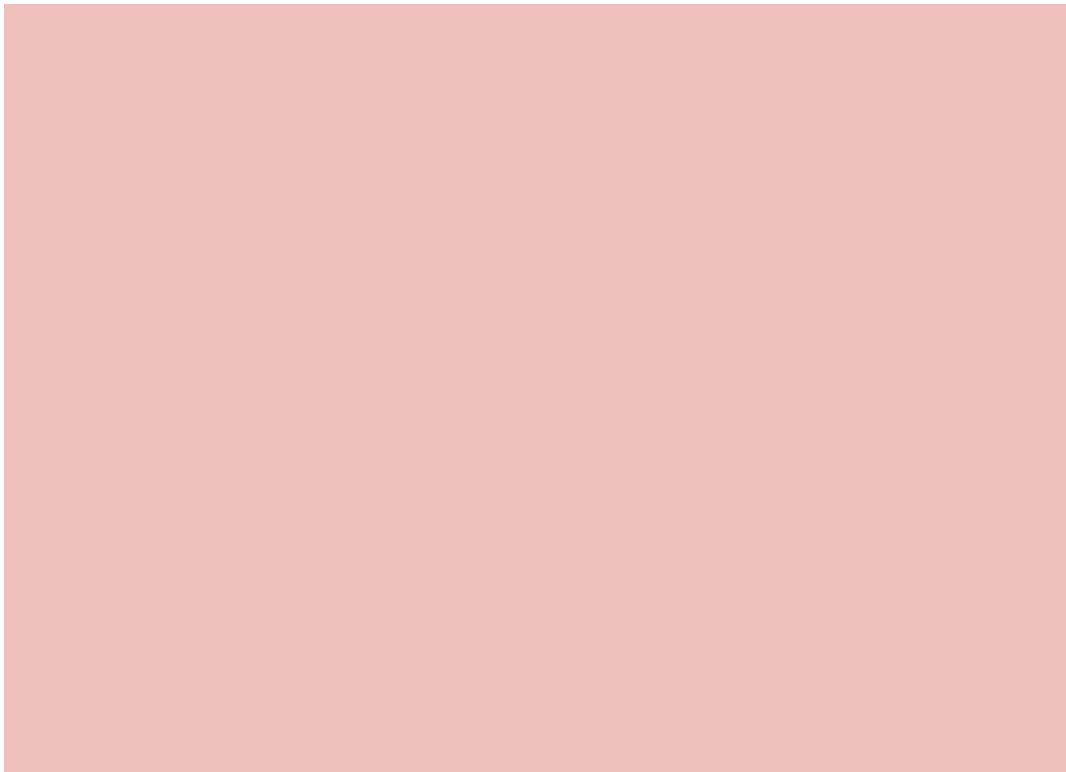
determinadas funcionalidades de nuestras páginas avanzadas. Además, sabiendo que dichas funcionalidades están cubiertas de una manera muy profesional y con cientos de horas de investigación que nosotros no deberíamos dedicar, a no ser que deseemos reinventar la rueda, que nunca es aconsejable.

Antes de acabar, apuntaremos la posibilidad de encontrar más directorios de scripts y programas listos para usar en general y de cualquier lenguaje, en nuestro [buscador en la categoría de colecciones de scripts](#). En el futuro y para continuar con esta exposición, comentaremos detenidamente alguno de estos programas de libre distribución para PHP.

Informe de **Miguel Angel Alvarez**
Director DesarrolloWeb.com
Mail: miguel@desarrolloweb.com

phpMyAdmin

phpMyAdmin es un programa de libre distribución en PHP, creado por una comunidad sin ánimo de lucro, que sólo trabaja en el proyecto por amor al arte. Es una herramienta muy completa que permite acceder a todas las funciones típicas de la base de datos MySQL a través de una interfaz web muy intuitiva.



La aplicación en si no es más que un conjunto de archivos escritos en PHP que podemos copiar en un directorio de nuestro servidor web, de modo que, cuando accedemos a esos archivos, nos muestran unas páginas donde podemos encontrar las bases de datos a las que tenemos acceso en nuestro servidor de bases de datos y todas sus tablas. La herramienta nos permite crear tablas, insertar datos en las tablas existentes, navegar por los registros de las tablas, editarlos y borrarlos, borrar tablas y un largo etcétera, incluso ejecutar sentencias SQL y hacer un backup de la base de datos.

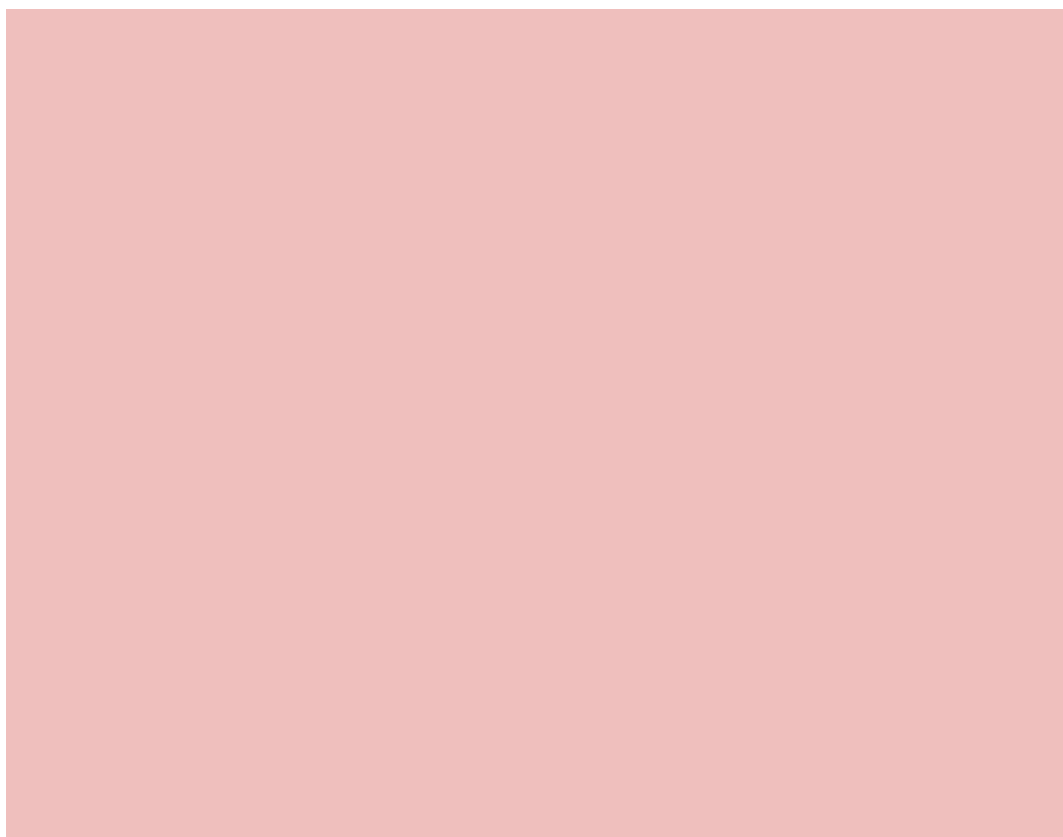
Página de phpMyAdmin

La página de inicio del proyecto es <http://www.phpmyadmin.net/>. Desde allí podemos descargar los ficheros de la última versión de la aplicación, que posteriormente debemos colocar en nuestro servidor web. También podemos encontrar a phpMyAdmin dentro de la red Sourceforge.net, que es un sitio que recoge multitud de proyectos "Open Source" (código abierto).

Hay varias versiones disponibles, pero es recomendable escoger la que nos aconsejen como la última versión estable (The last stable versión). En el momento de escribir este artículo era la 2.2.6. De modo que, si nuestro sistema es Windows, descargaremos el archivo phpMyAdmin-2.2.6-php.zip

Los archivos que hemos descargado son de la versión 4 de PHP, aunque también ofrecen la posibilidad de bajarse los archivos que guardan compatibilidad con la versión 3 de PHP, para que aquellos que no dispongan del motor de PHP más actual.

La pagina de inicio del programa también nos ofrece la posibilidad de ver un demo online, aunque nos avisan de que el servidor donde se aloja puede estar caído. <http://www.phpmyadmin.net/phpMyAdmin/>



Instalando phpMyAdmin

Una vez descargada la última versión la tenemos que descomprimir, con lo que obtendremos los ficheros PHP que conforman la herramienta y colocarlos dentro del directorio de publicación de nuestro servidor web.

Nota: recordamos que phpMyAdmin es un proyecto escrito en PHP, por lo que necesitaremos colocar los archivos en un servidor web que permita programación de páginas PHP. Además, deberemos acceder a la herramienta a través de la dirección del servidor web, seguida del directorio en el que tenemos los archivos que hemos descomprimido. Por ejemplo, si nuestro servidor es el [PWS](#) y hemos colocado los archivos dentro del directorio de publicación (Generalmente C:\Inetpub\wwwroot), en el subdirectorio phpMyAdmin, debemos escribir algo como `http://localhost/phpMyAdmin`

Si tuviéramos instalado un servidor Apache los colocaríamos en la carpeta que hayamos indicado como "documentRoot", que suele ser htdocs.

Lo primero que podemos leer es el archivo de la documentación, que encontramos junto con los archivos de phpMyAdmin. Explica datos generales del programa, como sus requerimientos, instrucciones de instalación, configuración, preguntas frecuentes, etc.

Posteriormente, tal como explica la documentación, hay que editar el archivo `config.inc.php` para cambiar los valores de host de la base de datos (ordenador que tiene instalado el MySQL) y el usuario y password con el que nos conectamos. Se pueden configurar muchos aspectos en la herramienta, aunque ahora solo comentaré los que he encontrado esenciales para hacerla funcionar, en la documentación tenemos un apartado dedicado por completo a especificar el sentido de cada variable.

\$cfgPmaAbsoluteUri

Debemos asignarlo a la ruta completa necesaria para acceder a phpMyAdmin. Podría ser algo como `http://localhost/phpMyAdmin` o `http://www.midominio.com/phpMyAdmin`

\$cfgServers[\$i]['host'] string

El nombre del host de la base de datos. Por ejemplo localhost, si es que es el mismo ordenador donde estamos instalando phpMyAdmin y la base de datos. También podría ser la dirección IP del ordenador al que nos conectamos.

\$cfgServers[\$i]['user'] string

\$cfgServers[\$i]['password'] string

El par usuario/contraseña que debe utilizar phpMyAdmin para conectarse con el servidor MySQL.

Con estas sencillas configuraciones ya podemos acceder a phpMyAdmin y trabajar con nuestra base de datos a golpe de ratón, que resulta muy de agradecer teniendo en cuenta que, en caso de no tener esta herramienta u otra parecida, la otra opción consistiría en utilizar el lenguaje SQL, y, en caso de que la base de datos esté alojada remotamente en Internet, no podríamos hacerlo sino es con acceso por TELNET al servidor de la base de datos.

Referencias: En DesarrolloWeb puedes conocer más cosas de PHP y MySQL.

En la [sección de PHP](#), podras aprender mucho de PHP, algo sobre MySQL e incluso sobre el lenguaje SQL.

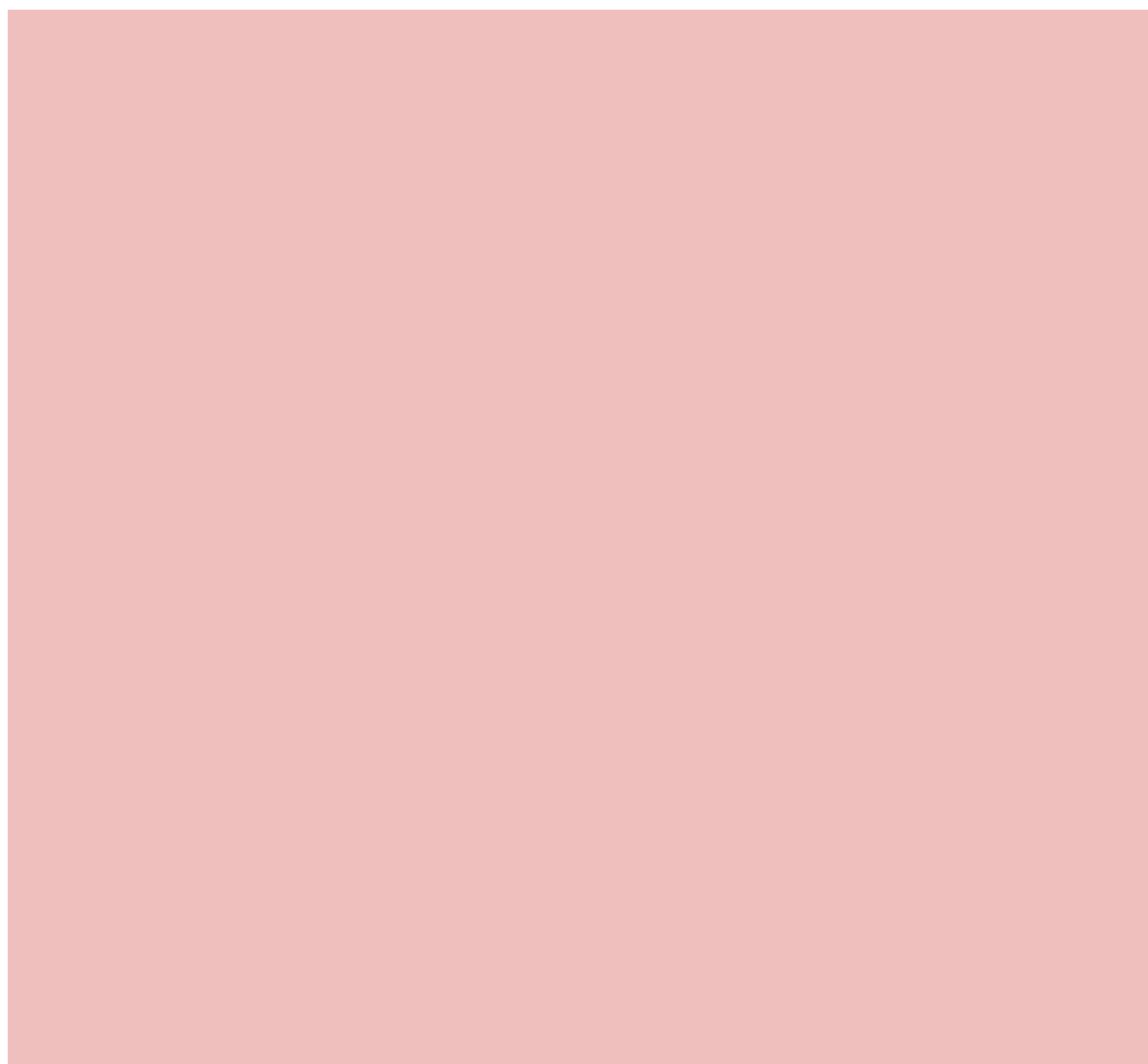
En el [directorio dedicado a MySQL](#) hay algunas referencias a artículos y enlaces externos.

Exportar datos de MySQL a Microsoft Access 2000

Migrar datos de una base de datos a otra es algo a lo que muchos de nosotros hemos tenido que confrontarnos en algún momento. A continuación os explicamos cómo recuperar información almacenada en un servidor de datos Mysql hacia una base Access 2000.

Referencia: Para realizar esta tarea es necesario que hayamos descargado el driver ODBC y lo hayamos instalado en nuestro sistema Windows. Esta labor se puede conocer en un artículo de DesarrolloWeb.com: [Instalar el driver ODBC para MySQL](#).

Para importar una tabla de Mysql a Microsoft Access, desde Access, y con la base de datos en la que se quieren importar los datos abierta, seleccionar el menu Archivo->Obtener datos Externos->Importar. En la pantalla de Importar datos, en la opción Tipo de archivo seleccionar ODBC databases().



Seleccionar origen de datos de equipo, y dentro de esta, el nombre de la fuente de datos que hemos creado anteriormente. Una vez la has seleccionado, y has hecho clic sobre "Aceptar", aparecerá la pantalla de configuración del driver por si deseas marcar para esta acción en concreto, algunas de las opciones de configuración que aparecen en el driver ODBC, si no deseas marcar ninguna, clic sobre "OK".

Nota: pudiera ser en algún caso que los tipos de los datos de la base en los sistemas MySQL y Access no sean totalmente compatibles y se produzca alguna anomalía al exportarlos. Realmente es una posibilidad que pensamos, aunque en las pruebas que hemos realizado no hemos visto ningún tipo de problema, bien es cierto que los campos que hemos trabajado no eran muy raros.

Aparecerá una ventana donde pregunta qué tabla de Mysql se desea exportar a Access:



Selecciona la tabla , y haz clic sobre "Aceptar"

Nota: si estamos exportando los datos hacia o desde un servidor de bases de datos alojado en algún proveedor de Hosting, tenemos que tener en cuenta que estos no siempre incluyen en su paquete básico el acceso remoto al servidor de base de datos, o requiere de un aviso explícito por parte del cliente para su configuración.

Referencia: si deseamos realizar una migración de datos en el otro sentido, es decir, desde Access hacia MySQL, será muy indicado leer otro artículo en DesarrolloWeb que explica el proceso detalladamente. [Exportar datos de Access 2000 a MySQL](#).

Informe de **Carlos Luis Cuenca**
Mail: carlos@desarrolloweb.com

Exportar datos de Access 2000 a MySQL

No es de extrañar que hayamos comenzado a hacer nuestros pinitos en la web sirviéndonos de una base de datos sencilla como Access. Tampoco es de extrañar que, llegado el momento, pasemos a cosas más serias y nos pasemos a un servidor de datos como MySQL. Aquí os mostramos una manera bastante práctica de migrar los datos de la una a la otra.

Referencia: Para realizar esta tarea es necesario que hayamos descargado el driver ODBC y lo hayamos instalado en nuestro sistema Windows. Esta labor se puede conocer en un artículo de DesarrolloWeb.com: [Instalar el driver ODBC para MySQL](#).

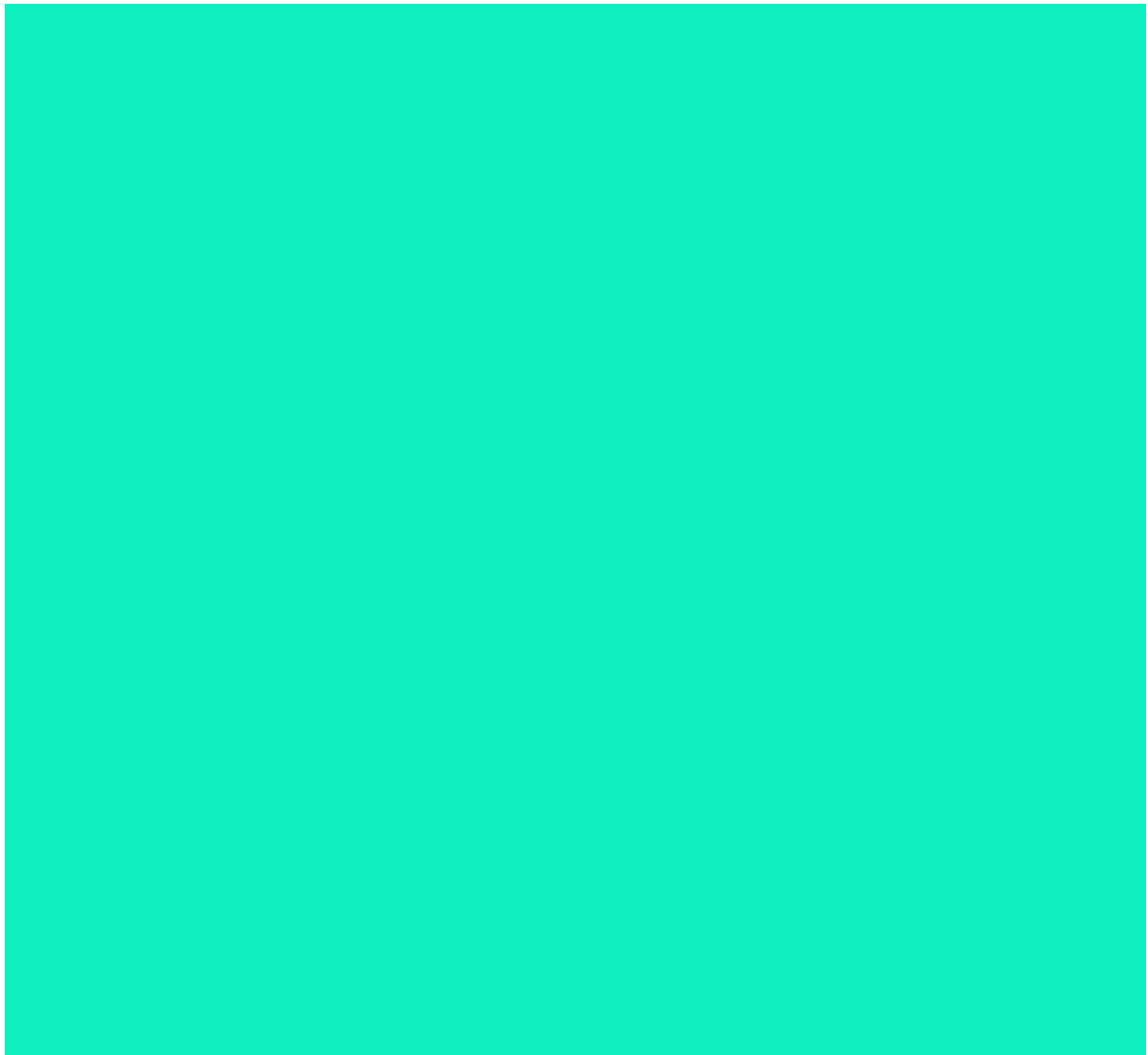
Para exportar una tabla a Mysql, hay que abrir la base de datos y seleccionar la tabla. Después, hacer clic sobre Archivo->Exportar. En la pantalla de exportar, en la opción Guardar como tipo, seleccionar ODBC databases().

Una vez se ha hecho esto, aparece una ventana que nos pregunta el nombre que le queremos dar a la tabla en Mysql, por defecto aparece el mismo.



Haz clic sobre "Aceptar", y aparecerá la pantalla en la que se pide que selecciones el origen de datos ODBC:

Nota: pudiera ser en algún caso que los tipos de los datos de la base en los sistemas MySQL y Access no sean totalmente compatibles y se produzca alguna anomalía al exportarlos. Realmente es una posibilidad que pensamos, aunque en las pruebas que hemos realizado no hemos visto ningún tipo de problema, bien es cierto que los campos que hemos trabajado no eran muy raros.



Seleccionar origen de datos de equipo, y dentro de esta el nombre de la fuente de datos que hemos creado anteriormente. Una vez la has seleccionado y has hecho clic sobre "Aceptar", aparecerá la pantalla de configuración del driver por si deseas marcar para esta acción en concreto algunas de las opciones de configuración que aparecen en el driver ODBC. Si no deseas marcar ninguna, haz clic sobre "OK" y los datos comenzarán a

exportarse.

Nota: si estamos exportando los datos hacia o desde un servidor de bases de datos alojado en algún proveedor de Hosting, tenemos que tener en cuenta que estos no siempre incluyen en su paquete básico el acceso remoto al servidor de base de datos, o requiere de un aviso explícito por parte del cliente para su configuración.

Referencia: si deseamos realizar una migración de datos en el otro sentido, es decir, desde MySQL hacia Access, será muy indicado leer otro artículo en DesarrolloWeb que explica el proceso detalladamente. [Exportar datos de MySQL a Microsoft Access 2000](#).

Informe de **Carlos Luis Cuenca**
Mail: carlos@desarrolloweb.com

Contador simple para páginas PHP

Hice una modificación al Script publicado en el artículo [Escritura de archivos con PHP](#), en el que se enseña a escribir archivos de texto mediante PHP, tocando los temas de lectura y escritura. El objetivo es llevar un conteo de las veces que se ha visitado una página.

Puse el siguiente script PHP al final de la página, se entenderá bien si se lee el artículo señalado antes.

```
<?
$archivo = "contador.txt";
$contador = 0;

$fp = fopen($archivo,"r");
$contador = fgets($fp, 26);
fclose($fp);

++$contador;

$fp = fopen($archivo,"w+");
fwrite($fp, $contador, 26);
fclose($fp);

echo "Esta página ha sido visitada $contador
veces";
?>
```

Además, creé un archivo llamado "contador.txt" que lo guardé en el mismo directorio que la página. Dicho archivo fue inicializado con un cero (0) como único texto.

Nota: si tenéis problemas a la hora de escribir en un archivo, casi con toda probabilidad, estará protegido contra escritura. O bien el archivo o bien el directorio.

Si tenéis vuestro propio servidor tendréis que modificar los permisos de tal archivo o directorio por vosotros mismos. Sin embargo, si estáis publicando en un alojamiento contratado en un proveedor tendréis que enteraros de qué mecanismo hay que poner en marcha en ese proveedor para conseguir los permisos. En muchos casos existirá un panel de control donde modificar esas cosas, en otros casos tendréis que escribir a soporte técnico para que lo hagan a mano ellos o os digan cómo hacerlo, si es que es posible.

Con esto ya está hecho un contador muy simple, pero muy funcional.

Referencia: Hemos publicado un artículo que amplía esta práctica sobre el contador. En concreto se hace un contador que no sólo registra las visitas, sino también las visitas en un mes y el mes de la última visita. El artículo se llama [Contador mejorado para páginas PHP](#).

También está publicado un artículo sobre [un contador PHP que utiliza imágenes](#) para mostrar el número de visitas.

Informe de **Daniel Guajardo**
Mail: daniel.guajardo@zoonico.cl

Gestión de directorios por PHP

Siguiendo con la saga de artículos referentes a la explotación de archivos por medio de PHP, vamos a presentar algunas funciones que nos pueden ser muy útiles en la navegación por directorios. Este tipo de funciones podrían, por ejemplo, servirnos para crear exploradores de archivos en nuestro navegador.

Funciones de gestión de directorios

Función	Descripción	Sintaxis
opendir	Abre un directorio situado en \$path y le asigna un identificador \$dir	\$dir = opendir(\$path)
readdir	Lee un elemento del directorio \$dir abierto previamente con opendir y desplaza el puntero al elemento siguiente	readdir(\$dir)
rmdir	Elimina el directorio \$dir	rmdir(\$dir)
mkdir	Crea un directorio situado en \$path con los derechos de acceso \$derechos (entero)	mkdir(\$path, \$derechos)
rewinddir	Vuelve el puntero de lectura del directorio \$dir al primer elemento	rewinddir(\$dir)
closedir	Cierra el directorio \$dir abierto previamente con opendir	closedir(\$dir)

La forma de tratar con estas funciones es la similar a la que ya hemos visto para la lectura secuencial de archivos. Podemos distinguir tres etapas elementales:

- Apertura del directorio por medio de la función *opendir* asignándole al mismo tiempo un identificador
- Realización de las tareas necesarias en relación con ese directorio
- Clausura del identificador por medio de la función *closedir*

A notar que, para que un directorio pueda ser borrado, hace falta previamente haber eliminado cada uno de los elementos contenidos dentro del directorio. Para esto, nos podemos servir de la función *unlink*, presentada en otro [artículo](#).

Por otra parte, la creación de un directorio por medio de la función *mkdir* requiere la definición de los derechos de acceso por medio de un numero entero. Esperamos poder explicar en qué consisten estos derechos próximamente.

Como ejemplo sencillo de lo que podemos hacer con estas funciones, aquí os presentamos un pequeño script de lectura que os permite visualizar el contenido de un directorio:

```
<?
//definimos el path de acceso
$path = "mi/camino";

//abrimos el directorio
$dir = opendir($path);

//Mostramos las informaciones
while ($elemento = readdir($dir))
{
    echo $elemento."<br>";
}

//Cerramos el directorio
closedir($dir);
?>
```

Otra forma de abordar la gestión de directorios es por medio de la clase *dir* que permite la creación de un objeto sobre el cual podremos aplicar toda una serie de métodos equivalentes a las funciones previamente vistas. Si estas familiarizado con la programación orientada a objetos, puede que esta modalidad te resulte mas intuitiva. Si no sabes en qué consiste la programación orientada a objetos, puedes visitar este [artículo](#).

En este caso, la forma de operar es análoga a la ya vista:

- Creamos un objeto *\$dir* con la instrucción: `$dir = dir($path)`
- Realizamos las tareas necesarias llamando a los métodos de la clase *dir*
- Cerramos el directorio con el método *close*

Algunos de los métodos que podemos utilizar con esta clase son los siguientes:

Métodos de la clase *dir*

Método	Descripción	Sintaxis
path	Indica el path del directorio	\$objeto->path
read	Lee un elemento del directorio	\$objeto->read
rewind	Vuelve el puntero de lectura del directorio al primer elemento	\$objeto->rewind
close	Cierra el directorio	\$objeto->close

Como ejemplo, he aquí el script equivalente al abordado para el caso de las funciones, esta vez usando la clase *dir*:

```
<?
//definimos el path de acceso
$path="mi/camino/";

//instanciamos el objeto
$dir=dir($path);

//Mostramos las informaciones
echo "Directorio ".$dir->path." : <br><br>";

while ($elemento = $dir->read())
{
    echo $elemento."<br>";
}
//Cerramos el directorio
$dir->close();
?>
```

Los scripts propuestos no son más que ejemplos sencillos de lo que estas funciones pueden ofrecernos. En vuestras manos queda el combinar estas funciones con otras vistas en este mismo taller de manera a crear aplicaciones que gestionen los archivos y directorios de vuestro servidor.

Recordamos que esta lista no es más que una recopilación y que muchas [otras funciones relacionadas](#) pueden sernos también útiles.

Informe de **Rubén Alvarez**
Mail: ruben@desarrolloweb.com

Mandar mails desde PHP

Para el envío de correos electrónicos utilizando PHP disponemos de una función bastante potente, incluida en todas las versiones de PHP, sin necesidad de instalar ningún añadido, en contra de lo que ocurría con ASP.

Referencia: En caso de que necesitemos programar el envío de correo electrónico en nuestra página utilizando ASP, también hemos publicado un artículo en DesarrolloWeb.com llamado [Mandar mails desde ASP](#).

En concreto, en PHP disponemos de una función llamada mail() que permite configurar y enviar el mensaje de correo. La función se llama mail() y recibe tres parámetros de manera obligada y otros dos parámetros que podemos colocar opcionalmente. Devuelve true si se envió el mensaje correctamente y false en caso contrario.

Parámetros necesarios en todos los casos

Destinatario: la dirección de correo o direcciones de correo que han de recibir el mensaje. Si incluimos varias direcciones debemos separarlas por una coma.

Asunto: para indicar una cadena de caracteres que queremos que sea el asunto del correo electrónico a enviar.

Cuerpo: el cuerpo del mensaje, lo que queremos que tenga escrito el correo.

Ejemplo de envío de un mail sencillo

```
<?
mail("pepito@desarrolloweb.com,maria@guiartemultimedia.com","asuntillo","Este es el cuerpo del
mensaje")
?>
```

Parámetros opcionales del envío de correo

Headers: Cabeceras del correo. Datos como la dirección de respuesta, las posibles direcciones que recibirán copia del mensaje, las direcciones que recibirán copia oculta, si el correo está en formato HTML, etc.

Additional_parameters: esta opción no suele utilizarse y, además, sólo está disponible a partir de la versión PHP 4.0.5 y desde PHP 4.2.3 está deshabilitado en modo seguro. Puede usarse para pasar parámetros adicionales al programa configurado para enviar el correo, cuando se manda el mail usando la opción de configuración `sendmail_path`. Podemos obtener más información en la [documentación de PHP para la función mail\(\)](#).

Ejemplo complejo de envío de correo

Vamos a enviar un correo con formato HTML a `pepito@desarrolloweb.com`, con copia a `mariano@desarrolloweb.com` y con copia oculta para `pepe@pepe.com` y `juan@juan.com`. La dirección de respuesta la configuraremos a `maria@desarrolloweb.com`.

```
<?
$destinatario = "pepito@desarrolloweb.com";
$asunto = "Este mensaje es de prueba";
$cuerpo = '
<html>
<head>
  <title>Prueba de correo</title>
</head>
<body>
<h1>Hola amigos!</h1>
<p>
<b>Bienvenidos a mi correo electrónico de prueba</b>. Estoy encantado de tener tantos lectores.
</p>
</body>
</html>
';

//para el envío en formato HTML
$headers = "MIME-Version: 1.0\r\n";
$headers .= "Content-type: text/html; charset=iso-8859-1\r\n";

//dirección del remitente
$headers .= "From: Miguel Angel Alvarez <pepito@desarrolloweb.com>\r\n";

//dirección de respuesta, si queremos que sea distinta que la del remitente
$headers .= "Reply-To: mariano@desarrolloweb.com\r\n";

//direcciones que recibían copia
$headers .= "Cc: maria@desarrolloweb.com\r\n";

//direcciones que recibirán copia oculta
$headers .= "Bcc: pepe@pepe.com,juan@juan.com\r\n";
```

```
mail($destinatario,$asunto,$cuerpo,$headers)
?>
```

Nota: Antes de poner en marcha el script en vuestro servidor, por favor, cambiar los datos de configuración de las direcciones de correo que van a recibir el mensaje y colocar unas direcciones que sean vuestras y donde podáis comprobar si los mensajes se envían correctamente.

Conclusión y descarga

Pensamos y esperamos que después de este artículo compartáis nuestra opinión, que el envío de mails en PHP es una tarea muy sencilla. Además, es muy de agradecer que todas las versiones de PHP incluyan una función para el envío de mails.

Ponemos a vuestra disposición para la descarga el archivo .php con el [código anterior completo para realizar el envío de mails](#).

Nota: Para el envío de correo mediante PHP es necesario que este disponga de una correcta configuración.

Si nuestro web está en un servidor de un proveedor de hosting seguramente ya hayan configurado PHP para el envío de mails. Si estamos trabajando en un servidor propio, sí tendremos que configurar PHP.

PHP se configura en el archivo php.ini, donde debemos especificar datos como el servidor de correo saliente que debe de utilizar PHP para transferir los mensajes.

Dentro del php.ini, debemos buscar el epígrafe [mail function]. Dependiendo de nuestro sistema deberemos configurar de una manera u otra.

En sistemas Windows encontraremos el php.ini en el directorio windows o dentro de este, en el subdirectorio system32 o similar. En este sistema deberemos indicar el dominio del servidor de smtp, algo como smtp.midominio.com. Si es el ordenador local el que hace de servidor, podremos poner "localhost" como máquina que enviará el correo. También podemos especificar la dirección desde donde queremos que parezca que se envía el mensaje en caso de que no se indique otra durante el envío.

Informe de **Miguel Angel Alvarez**
Director DesarrolloWeb.com
Mail: miguel@desarrolloweb.com

validar email en PHP

Vamos a ver una función muy útil en PHP que sirve para comprobar la validez de un correo. En realidad comprueba si una dirección de correo electrónico está bien escrita sintácticamente, dejando de lado las comprobaciones de si ese mail existe o no realmente, que no se pueden hacer tan fácilmente.

Vamos a escribir una función que se llama comprobar_email y recibe la cadena de texto con el email que queremos validar. Si dicho email es correcto desde el punto de vista sintáctico, es decir, si tiene un nombre de usuario, una arroba y una terminación con el nombre de un dominio o subdominio, etc, devolverá un 1, es decir, verdadero. En caso de que el email no esté correctamente escrito, la función devolvería 0, que equivale a falso.

La función en si da por hecho inicialmente que el email es erróneo y realiza una serie de

comprobaciones que, si todas responden correctamente, dan por conclusión que el email sí estaba bien escrito. Si alguna de esas comprobaciones no era correcta, no se llegaría al final de las comprobaciones y quedaría el resultado como se ha supuesto en un principio, es decir, como incorrecto.

código de la función

```
function comprobar_email($email){
    $mail_correcto = 0;
    //compruebo unas cosas primeras
    if ((strlen($email) >= 6) && (substr_count($email,"@") == 1) && (substr($email,0,1) != "@") && (substr($email,strlen($email)-1,1) != "@")){
        if ((!strstr($email,"")) && (!strstr($email,"\"")) && (!strstr($email,"\\")) && (!strstr($email,"\$")) && (!strstr($email," "))) {
            //miro si tiene caracter .
            if (substr_count($email,".") >= 1){
                //obtengo la terminacion del dominio
                $term_dom = substr(strrchr ($email, '.'),1);
                //compruebo que la terminación del dominio sea correcta
                if (strlen($term_dom)>1 && strlen($term_dom)<5 && (!strstr($term_dom,"@")) ){
                    //compruebo que lo de antes del dominio sea correcto
                    $antes_dom = substr($email,0,strlen($email) - strlen($term_dom) - 1);
                    $caracter_ult = substr($antes_dom,strlen($antes_dom)-1,1);
                    if ($caracter_ult != "@" && $caracter_ult != "."){
                        $mail_correcto = 1;
                    }
                }
            }
        }
    }
    if ($mail_correcto)
        return 1;
    else
        return 0;
}
```

Las comprobaciones

En el primer if compruebo que el email tiene por lo menos 6 caracteres (el mínimo), que tiene una arroba y sólo una y que no está colocada ni al principio ni al final.

En el segundo if comprueba que no tiene algunos caracteres no permitidos. Y los restantes hacen comprobaciones de las distintas partes de la dirección de correo, a saber: Que hay un punto en algún lado y que la terminación del dominio es correcta y que el principio de la dirección también es correcto.

Finalmente, se devuelve la variable local utilizada para guardar la validez o incorrección del correo.

Descarga del script

La persona que lo desee, puede [descargar el archivo con el código fuente de este script](#), para utilizarlo en sus aplicaciones web.

Informe de **Miguel Angel Alvarez**
Director DesarrolloWeb.com
Mail: miguel@desarrolloweb.com

Paginación de resultados con PHP y MySQL

En muchas ocasiones, cuando se presentan en una página web registros de una base de datos, se deberían mostrar demasiados registros como para colocarlos todos en una única página. En estas ocasiones se suele paginar los resultados, quizás cientos, en distintas páginas con conjuntos de registros mucho menos numerosos. Por ejemplo, podríamos presentar los resultados en páginas de 10 elementos o 20, dependiendo de nuestras intenciones y el tipo de datos que se estén presentando. Este efecto lo habremos podido observar repetidas veces en los buscadores.

Podríamos desarrollar distintos scripts para paginar resultados en PHP. En este artículo vamos a explicar una posibilidad basada en la utilización de una base de datos MySQL y sentencias SQL a las que indicaremos el conjunto de registros que queremos mostrar en cada página. Los enunciados SELECT del lenguaje SQL, en la base de datos MySQL y otras muchas, tienen una cláusula llamada LIMIT, con la que podemos indicar los registros a mostrar, por ejemplo, 10 registros empezando por el registro 180.

```
select * from pais limit 180,10
```

Como vemos LIMIT tiene dos argumentos, el primero es el registro por el que empezar los resultados y el segundo el número de resultados a recoger en el conjunto de registros resultante.

Así pues, en este ejercicio de paginación la cláusula LIMIT será la clave para mostrar los registros en grupos del tamaño deseado.

Podemos [ver el resultado que vamos a conseguir en este artículo](#) ahora y así tendremos más facilidad de identificar las distintas partes del código que vamos a comentar.

Código de paginación

Hay varias partes del código que servirán específicamente para implementar la paginación. Lo primero es saber qué página se desea mostrar. En principio se mostraría la primera página de resultados, pero si un visitante selecciona con los enlaces de abajo otra página distinta de la primera, habría que mostrarla también. El índice de la página a mostrar, si es que no es la primera vez que se accede, se recibe por parámetro en la URL.

```
//Limito la busqueda
$TAMANO_PAGINA = 10;

//examinio 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;
}
```

Estoy definiendo el tamaño de la página. Luego procuro recibir la página por parámetro en la URL. Si no se recibió nada, se entiende que la página a mostrar es la primera, luego la variable \$inicio, que guarda el primer registro a mostrar (para indicarlo en la sentencia SQL en el apartado LIMIT), será cero. Si habíamos recibido algo como página, calculo el inicio con una simple multiplicación de la página a mostrar por el tamaño_ de página

definido antes.

Es habitual en estas páginas de resultados informar un poco sobre la cantidad de registros encontrados y los datos de la página que estamos viendo. Estos datos se pueden obtener con unas sencillas operaciones.

```
//miro a ver el número total de campos que hay en la tabla con esa búsqueda
$$sql = "select * from pais " . $criterio;
$rs = mysql_query($$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>";
```

Nota: Este código podría mostrar una información como esta:

```
Número de registros encontrados: 256
Se muestran páginas de 10 registros cada una
Mostrando la página 19 de 26
```

Lo primero es hacer una búsqueda en la base de datos por el criterio que se esté utilizando para saber cuantos registros se obtienen en total sin la paginación (luego veremos de donde sale la variable \$criterio).

A continuación puedo calcular el número total de páginas de resultados que genera la búsqueda. La función ceil() redondea números en coma flotante o reales hacia arriba, así pues, devuelve el entero por arriba más próximo.

Las siguientes líneas, donde se utiliza echo, tienen como objeto mostrar los datos en la página.

Ahora veremos el código que realiza la búsqueda en la base de datos, extrayendo y mostrando solamente aquellos registros que corresponden con la página a mostrar.

```
//construyo la sentencia SQL
$$sql = "select * from pais " . $criterio . " limit " . $inicio . "," . $TAMANO_PAGINA;
$rs = mysql_query($$sql);
while ($fila = mysql_fetch_object($rs)){
    echo $fila->nombre_pais . "<br>";
}
//cerramos el conjunto de resultado y la conexión con la base de datos
mysql_free_result($rs);
mysql_close($conn);
```

Se construye la sentencia SQL para extraer los datos con el criterio, que veremos luego de donde sale, pero que en principio lo podemos tomar como una cadena vacía. También se utiliza LIMIT, como ya se indicó: poniendo los valores definidos antes como inicio y tamaño de página.

El resto es un recorrido típico por un conjunto de registros, en este caso los países de nuestra base de datos, donde se van mostrando todos los elementos desde el principio hasta el final. Finalizando este recorrido no vamos a realizar ninguna acción más con la base de datos, así que podemos cerrar el resultado de la búsqueda y la conexión con la base de datos.

Ahora podemos ver el código que muestra en la parte de abajo los numeritos de todas las páginas que genera la búsqueda, para que el visitante pueda seleccionar una página y moverse entre los resultados.

```
//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 actualmente, coloco el enlace para ir a esa página
            echo "<a href='index.php?pagina=" . $i . "&criterio=" . $txt_criterio . "'>" . $i . "</a> ";
    }
}
```

La primera línea comprueba si realmente hay varias páginas de resultados, pues, si no es así no tendría que mostrar nada. Si efectivamente hay varias páginas para mostrar se recorren todas y para cada una se muestra el índice.

El índice a mostrar puede que sea el de la página que se está visualizando en ese momento y en ese caso simplemente podríamos el numerito, pero no el enlace para ir a ese documento, pues es en el que estamos. En caso de que sea una página de resultados distinta, se muestra un enlace para moverse a dicha página, donde se incluye pasando por parámetro tanto el índice de la página que se desea ver como el criterio de la búsqueda que se estaba realizando.

Hasta aquí el código imprescindible para la paginación. Aunque aun vamos a ver alguna cosa más.

Código de búsqueda

Para hacer un taller un poco más completo y poder ofrecer una página de muestra con funcionalidades de búsqueda, hemos creado la posibilidad de añadir un criterio para encontrar tan sólo elementos relacionados con él. Después de todo, los códigos de paginación suelen utilizarse en situaciones en las que se están realizando búsquedas en la base de datos.

El criterio se podrá definir en una caja de texto y habrá un botón de buscar que llame a la misma página pero pasando el texto con las palabras a buscar en la base de datos.

```
<form action="index.php" method="get">
Criterio de búsqueda:
<input type="text" name="criterio" size="22" maxlength="150">
<input type="submit" value="Buscar">
</form>
```

Nos fijamos que el método por el que pasamos este formulario es GET. Esto es debio a que no queremos liar el código y como estamos pasando ya por GET el criterio en otros sitios, utilizamos el mismo método.

El formulario lo colocaremos debajo, pero habrá también un trozo de código que recogerá la información y la tratará para adaptarla a una sentencia de búsqueda en la base de datos. Este código lo colocaremos en la parte de arriba de la página.

```
//inicializo el criterio y recibo cualquier cadena que se desee buscar
```

```
$criterio = "";  
if ($_GET["criterio"]!=""){  
    $txt_criterio = $_GET["criterio"];  
    $criterio = " where nombre_pais like '%" . $txt_criterio . "%'";  
}
```

Se inicializa el criterio a una cadena vacía y luego se comprueba si se ha recibido algo por método GET en el campo criterio. Si era así se puede recoger el texto recibido y construir el criterio, que no es más que una cláusula WHERE donde se buscan elementos, en este caso países, cuyo nombre contenga por algún sitio las letras que se han recibido como texto del criterio.

Base de datos

Como se dijo, se está utilizando una base de datos MySQL. En el [Manual de Programación en PHP de DesarrolloWeb.com](#) se muestra la manera de trabajar con bases de datos.

En nuestro ejemplo nos faltan por indicar las sentencias para conectar con MySQL y seleccionar la base de datos a utilizar. Serían unas parecidas a estas.

```
//conecto con la base de datos  
$conn = mysql_connect("servidor","usuario","password");  
mysql_select_db("nombre_bbdd",$conn);
```

Conclusión

Para acabar, ponemos a vuestra disposición la descarga del [código de este ejercicio](#), donde se puede ver todo el ejemplo completo y comentado. Esperamos que podáis aplicarlo a vuestros desarrollos.

Referencia: A posteriori, hemos recibido [un código para paginación realizado por un lector](#), que nos lo ha mandado para complementar este artículo. No podemos ofrecer los créditos porque hemos perdido su comunicación, así que si alguien lo reivindica colocaremos la autoría gustosamente.

Asimismo, disponemos de otro artículo que trata la paginación de resultados en PHP con un enfoque un poco distinto. El artículo viene con el script explicado y todo lo necesario para ponerlo en marcha: [Paginación PHP y MySQL. Ejemplo 2](#).

Informe de **Miguel Angel Alvarez**
Director DesarrolloWeb.com
Mail: miguel@desarrolloweb.com

Paginacion con PHP y MySQL. Ejemplo 2

La paginación de resultados con PHP es uno de los talleres más interesantes y prácticos que se pueden hacer con esta tecnología. Casi todos los sitios web que implementan algún tipo de búsqueda deben en algún momento utilizar la paginación de resultados para no arrojar una cantidad desmesurada de datos en una única página. Como se ha visto desde siempre en los buscadores tipo Google, es mucho mejor presentar los resultados en varias páginas distintas y colocar en la parte de abajo una lista de las páginas de resultados que la búsqueda ha encontrado.

Para realizar este ejercicio se ha utilizado una base de datos MySQL y programación utilizando la tecnología PHP.

Podemos [ver el resultado que vamos a conseguir con este artículo](#) ahora y así tendremos más facilidad de identificar las distintas partes del código que vamos a comentar.

Nota: La paginación de resultados ya se ha [visto en un artículo anterior de PHP](#). En este caso ofrecemos otro ejemplo realizado por otro programador que seguro que nos sirve para hacernos una idea más global sobre como afrontar este tipo de problemas.

El código de este ejercicio, junto con algunas instrucciones para ponerlo en marcha, se puede [descargar en este enlace](#).

Poner en marcha el ejemplo

En el propio código de la paginación se encuentran las instrucciones para ponerlo en marcha y explicaciones sobre el funcionamiento del script. En este texto se reproducen algunas de las notas ofrecidas para ponerlo en marcha.

- 1.- Copiar el fichero busqueda.php en cualquier directorio del servidor web
- 2.- Crear una base de datos llamado 'tpv' (si es diferente, deberíais cambiar el código)
- 3.- Cambiar los datos de conexión si son diferentes.
- 4.- Cargar la siguiente tabla en la base de datos:

```
CREATE TABLE comercios (  
  co_id varchar(10) NOT NULL default "",  
  co_nombre varchar(30) NOT NULL default "",  
  co_pais varchar(30) NOT NULL default "",  
  UNIQUE KEY co_id (co_id)  
) TYPE=MyISAM;
```

- 5.- Realizar los siguientes insert

```
INSERT INTO comercios VALUES ('ESGR000002', 'PRUEBA 2', 'ESPAÑA');  
INSERT INTO comercios VALUES ('ESGR000001', 'PRUEBA 1', 'ESPAÑA');  
INSERT INTO comercios VALUES ('516', 'JUAN PEREZ', 'MEXICO');  
INSERT INTO comercios VALUES ('984', 'ANTONIO RODRIGUEZ', 'MEXICO');  
INSERT INTO comercios VALUES ('996', 'INDALECIO TRAVIANNI', 'ARGENTINA');  
INSERT INTO comercios VALUES ('975', 'ABELARDO SAINZ', 'PERU');  
INSERT INTO comercios VALUES ('111', 'JOSE TOLTACA', 'PERU');  
INSERT INTO comercios VALUES ('332', 'RAIMUNDO ALONSO', 'ARGENTINA');  
INSERT INTO comercios VALUES ('123', 'JUN JUANES', 'BRASIL');  
INSERT INTO comercios VALUES ('585', 'JOAO PAMINHIO', 'BRASIL');  
INSERT INTO comercios VALUES ('23432432', 'JOAQUIN DIAZ', 'ESPAÑA');
```

Por supuesto estos datos pueden ser los que queráis, solo sirve de ejemplo.

Una vez realizados todos estos pasos, ejecutar el fichero busqueda.php... y a probar... :D

Nota: Al ser este un pequeño ejemplo, en el formulario de búsqueda he dejado que busque el valor introducido en todos los campos.

El código del script se puede ver a continuación:

```
<?  
mysql_connect("localhost","root","");  
?>
```



```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>ejemplo de paginación de resultados</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<meta http-equiv="Pragma" content="no-cache" />
<style type="text/css">
<!--
a.p:link {
    color: #0066FF;
    text-decoration: none;
}
a.p:visited {
    color: #0066FF;
    text-decoration: none;
}
a.p:active {
    color: #0066FF;
    text-decoration: none;
}
a.p:hover {
    color: #0066FF;
    text-decoration: underline;
}
a.ord:link {
    color: #000000;
    text-decoration: none;
}
a.ord:visited {
    color: #000000;
    text-decoration: none;
}
a.ord:active {
    color: #000000;
    text-decoration: none;
}
a.ord:hover {
    color: #000000;
    text-decoration: underline;
}
-->
</style>
</head>
<body bgcolor="#FFFFFF">
<script language="JavaScript">
function muestra(queCosa)
{
    alert(queCosa);
}
</script>
<div align="center"><strong><font color="#000000" size="2" face="Verdana, Arial, Helvetica, sans-serif">Paginación
de Resultados de una consulta SQL (sobre MySQL)<br><br><p><a href="http://www.pclandia.com">www.
pclandia.com</a></p> </font></strong> </div>
<hr noshade style="color:CC6666;height:1px">
<br>
<?
//inicializo el criterio y recibo cualquier cadena que se desee buscar
$criterio = "";
$txt_criterio = "";
if ($_GET["criterio"]!=""){
    $txt_criterio = $_GET["criterio"];
    $criterio = " where co_id like '%" . $txt_criterio . '%" or co_nombre like '%" . $txt_criterio . '%" or co_pais
like '%" . $txt_criterio . "%'";
}

```

```

$sql="SELECT * FROM tpv.comercios ".$criterio;
$res=mysql_query($sql);
$numeroRegistros=mysql_num_rows($res);
if($numeroRegistros<=0)
{
    echo "<div align='center'>";
    echo "<font face='verdana' size='-2'>No se encontraron resultados</font>";
    echo "</div>";
}else{
    //////////elementos para el orden
    if(!isset($orden))
    {
        $orden="co_id";
    }
    //////////fin elementos de orden

    //////////calcula de elementos necesarios para paginacion
    //tamaño de la pagina
    $tamPag=5;

    //pagina actual si no esta definida y limites
    if(!isset($_GET["pagina"]))
    {
        $pagina=1;
        $inicio=1;
        $final=$tamPag;
    }else{
        $pagina = $_GET["pagina"];
    }
    //calcula del limite inferior
    $limitInf=($pagina-1)*$tamPag;

    //calcula del numero de paginas
    $numPags=ceil($numeroRegistros/$tamPag);
    if(!isset($pagina))
    {
        $pagina=1;
        $inicio=1;
        $final=$tamPag;
    }else{
        $seccionActual=intval(($pagina-1)/$tamPag);
        $inicio=($seccionActual*$tamPag)+1;

        if($pagina<$numPags)
        {
            $final=$inicio+$tamPag-1;
        }else{
            $final=$numPags;
        }

        if ($final>$numPags){
            $final=$numPags;
        }
    }
}

//////////fin de dicho calculo

//////////creacion de la consulta con limites
$sql="SELECT * FROM tpv.comercios ".$criterio." ORDER BY ".$orden.",co_id ASC LIMIT ".$limitInf.", ".$tamPag;
$res=mysql_query($sql);

//////////fin consulta con limites
echo "<div align='center'>";
echo "<font face='verdana' size='-2'>encontrados ".$numeroRegistros." resultados<br>";

```

```

echo "ordenados por <b>".$orden."</b>";
if(isset($txt_criterio)){
    echo "<br>Valor filtro: <b>".$txt_criterio."</b>";
}
echo "</font></div>";
echo "<table align='center' width='80%' border='0' cellspacing='1' cellpadding='0'>";
echo "<tr><td colspan='3'><hr noshade></td></tr>";
echo "<th bgcolor='#CCCCCC'><a class='ord' href='\"".$_SERVER["PHP_SELF"]."\"?pagina = ".$pagina."&orden=co_id&criterio=".$txt_criterio.">Código</a></th>";
echo "<th bgcolor='#CCCCCC'><a class='ord' href='\"".$_SERVER["PHP_SELF"]."\"?pagina = ".$pagina."&orden=co_nombre&criterio=".$txt_criterio.">Nombre</a></th>";
echo "<th bgcolor='#CCCCCC'><a class='ord' href='\"".$_SERVER["PHP_SELF"]."\"?pagina = ".$pagina."&orden=co_pais&criterio=".$txt_criterio.">País</a></th>";
while($registro=mysql_fetch_array($res))
{
    <?
    <!-- tabla de resultados -->
    <tr bgcolor="#CC6666" onMouseOver="this.style.backgroundColor='#FF9900';this.style.cursor='hand';"
onMouseOut="this.style.backgroundColor='#CC6666'"o";" onClick="javascript: muestra(' <? echo "[".$registro
["co_id"]."] ".$registro["co_nombre"]." - ".$registro["co_pais"]; ?>');">
    <td><font size="2" face="Verdana, Arial, Helvetica, sans-serif" color="#FFFFCC"><b><? echo $registro
["co_id"]; ?></b></font></td>
    <td><font size="2" face="Verdana, Arial, Helvetica, sans-serif" color="#FFFFCC"><b><? echo $registro
["co_nombre"]; ?></b></font></td>
    <td><font size="2" face="Verdana, Arial, Helvetica, sans-serif" color="#FFFFCC"><b><? echo $registro
["co_pais"]; ?></b></font></td>
    </tr>
    <!-- fin tabla resultados -->
    <?
} //fin while
echo "</table>";
} //fin if
//////////a partir de aqui viene la paginacion
<?
    <br>
    <table border="0" cellspacing="0" cellpadding="0" align="center">
    <tr><td align="center" valign="top">
    <?
        if($pagina>1)
        {
            echo "<a class='p' href='\"".$_SERVER["PHP_SELF"]."\"?pagina= ".$pagina-1."&orden=".$orden."&criterio=".$txt_criterio.">";
            echo "<font face='verdana' size='-2'>anterior</font>";
            echo "</a> ";
        }

        for($i=$inicio;$i<=$final;$i++)
        {
            if($i==$pagina)
            {
                echo "<font face='verdana' size='-2'><b>".$i."</b> </font>";
            }else{
                echo "<a class='p' href='\"".$_SERVER["PHP_SELF"]."\"?pagina= ".$i."&orden=".$orden."&criterio=".$txt_criterio.">";
                echo "<font face='verdana' size='-2'>".$i."</font></a> ";
            }
        }
        if($pagina<$numPags)
        {
            echo " <a class='p' href='\"".$_SERVER["PHP_SELF"]."\"?pagina= ".$pagina+1."&orden=".$orden."&criterio=".$txt_criterio.">";
            echo "<font face='verdana' size='-2'>siguiente</font></a>";
        }
    }
    //////////fin de la paginacion
    <?

```

```

</td></tr>
</table>
<hr noshade style="color: CC6666; height: 1px">
<div align="center"><font face="verdana" size="-2"><a class="p" href="index.php">:: Inicio: </a></font></div>

<form action="busqueda.php" method="get">
Criterio de búsqueda:
<input type="text" name="criterio" size="22" maxlength="150">
<input type="submit" value="Buscar">
</form>

</body>
</html>
<?
mysql_close();
?>

```

Se puede [descargar aquí](#).

Informe de **Francisco J. Matias**
 Mail: PCLANDIA@terra.es
 URL: <http://www.pclandia.com>

Biblioteca ADOdb para PHP

Esta nota técnica la envío a propósito de que ayer leí un taller de acceso a BD con PHPLIB, el cual me pareció muy interesante y creo necesario ampliar las ventajas de PHP utilizando la biblioteca ADODB.

La forma de como instalar los archivos y de las especificaciones, las cuales son muy muy sencillas, se las dejo a ustedes, pueden encontrar el manual y la descarga en: <http://php.weblogs.com/>

El asunto aquí es que adodb maneja objetos, al igual que PHPLIB, pero lo fantástico es que solo bastará que hagamos una sola vez nuestro código sin importar si nuestro DBMS cambia, así nos evitaremos el tener que mudar el código también y no haremos un sitio para cada manejador, pues el tipo de BD lo especificamos en una variable y tan solo hará falta cambiar esa variable por el nombre del actual DBMS.

Aquí les pondré tan solo un pequeño ejemplo de manejo de datos con ADODB.

```

<?
include('adodb.inc.php'); //Este es un archivo con los datos que necesita el programa para reconocer adodb

$conn = &ADONewConnection('mysql'); Aquí el tipo de BD
$conn->PConnect('usuarios'); Conexión con la BD

if (!$DB)
    print "No se realizó la conexión";
else{
    $query= "select * from usuario";
    $query.= "where nombre like '%$nombre%'";

    $datos = $DB->Execute("$query"); /*Ejecutamos el query*/

    $numFilas = $datos->RecordCount(); /* contamos el total de registros de resultado */

    for ($i=0; $i<=$numFilas; $i++){

```

```

/* Comenzamos a extraer de la BD los registros */
$nombre= $datos->fields["nombre"];
$appat = $datos->fields["apellidoP"];

print " $i. Nombre:$nombre $appat";
}

$datos->Close(); //opcional
$conn->Close(); //opcional
}
?>

```

Bueno, ahí el pequeño ejemplo, ya les toca a ustedes investigar y aprovechar la biblioteca a como les acomode, saludos!

Informe de **Roberto Bárcenas**

Mail: yaca_13@hotmail.com

URL: <http://www.chicosyescritores.org>

Enviar un formulario por mail con PHP

El método para enviar un formulario con PHP resulta muy similar al [utilizado en ASP](#). Varía tan sólo la sintaxis utilizada y las líneas de código que realizan el envío del correo electrónico.

Para empezar, sería muy útil que [aprendamos a enviar correos electrónicos con PHP](#), para lo que tenemos un artículo en DesarrolloWeb.

Esquema de funcionamiento

En este caso nos vamos a apoyar en la variable `$HTTP_POST_VARS`, que debería contener el formulario, para saber si hemos recibido o no datos desde un formulario. Dicha variable la utilizamos en un enunciado `if (! $HTTP_POST_VARS)`, que si pasa por el caso positivo -no había nada en `$HTTP_POST_VARS`- significa que no se ha recibido nada desde un formulario. En ese caso, muestro el formulario de contacto.

En caso contrario -sí que había algo en `$HTTP_POST_VARS`- quiere decir que estamos recibiendo datos por un formulario y en ese caso, recogemos los datos y componemos el cuerpo del mensaje.

Veamos el código de la página para crear el formulario, recibirlo y componer el cuerpo del mensaje que se va a enviar. En el mismo código podremos encontrar también la llamada a la función que envía el correo.

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
  <title>Mándanos tus comentarios</title>
</head>

<body bgcolor="#cccc66" text="#003300" link="#006060" vlink="#006060">
<?
if (! $HTTP_POST_VARS){
?>
<form action="envia_form_php.php" method=post>
Nombre: <input type=text name="nombre" size=16>

```

```

<br>
Email: <input type=text name=email size=16>
<br>
Comentarios: <textarea name=coment cols=32 rows=6></textarea>
<br>
<input type=submit value="Enviar">
</form>
<?
}else{
    //Estoy recibiendo el formulario, compongo el cuerpo
    $cuerpo = "Formulario enviado\n";
    $cuerpo .= "Nombre: " . $HTTP_POST_VARS["nombre"] . "\n";
    $cuerpo .= "Email: " . $HTTP_POST_VARS["email"] . "\n";
    $cuerpo .= "Comentarios: " . $HTTP_POST_VARS["coment"] . "\n";

    //mando el correo...
    mail("admin@tudominio.com","Formulario recibido",$cuerpo);

    //doy las gracias por el envío
    echo "Gracias por rellenar el formulario. Se ha enviado correctamente.";
}
?>
</body>
</html>

```

Informe de **Miguel Angel Alvarez**
 Director DesarrolloWeb.com
 Mail: miguel@desarrolloweb.com

Cálculo de los días de un mes en PHP

A continuación vamos a ver una sencilla manera de hacer una función en PHP que realiza el cálculo de los días de un mes. Es la función UltimoDia() que hemos utilizado ya en alguna ocasión para el manual de [calendario en PHP](#).

Nota: La función UltimoDia() hace un cálculo de cuál es el último día de un mes. La hemos utilizado con anterioridad (una versión distinta de la presente) para realizar el [Manual del calendario en PHP](#).

No es necesario hacer un ciclo repetitivo para la función UltimoDia(), es un poco más simple de resolver.

Los meses 1,3,5,7,8,10,12 siempre tienen 31 días, los meses 4,6,9,11 siempre tienen 30 días, el único problema es el mes de febrero dependiendo del año puede tener 28 o 29 días, pero ese cálculo tampoco es difícil.

Aquí envío el código para la función UltimoDia(), que ojalá les sirva...

```

function UltimoDia($anho,$mes){
    if (((fmod($anho,4)==0) and (fmod($anho,100)!=0)) or (fmod($anho,400)==0)) {
        $dias_febrero = 29;
    } else {
        $dias_febrero = 28;
    }
    switch($mes) {
        case 01: return 31; break;
        case 02: return $dias_febrero; break;
        case 03: return 31; break;
        case 04: return 30; break;
    }
}

```

```
case 05: return 31; break;
case 06: return 30; break;
case 07: return 31; break;
case 08: return 31; break;
case 09: return 30; break;
case 10: return 31; break;
case 11: return 30; break;
case 12: return 31; break;
```

```
}
}
```

Informe de **Héctor A. Pinto F**
Ingeniero en Informática. Santiago de Chile.
Mail: hpinto@tutopia.com

Problemas con las fechas en timestamp Unix de PHP

Existen en PHP una serie de funciones muy útiles para realizar cálculos de fechas, como por ejemplo, saber si una fecha es válida, obtener un dato concreto de una fecha, como el día, la hora o el día del mes o la semana.

En teoría, cualquier cálculo con fechas básico se puede realizar con las funciones que provee el lenguaje, que se pueden observar en la documentación de PHP, concretamente en la URL <http://www.php.net/manual/es/ref.datetime.php>

El problema del Timestamp Unix

Muchas de las funciones de PHP se basan en el Timestamp de Unix que es el número de segundos transcurridos desde las 00:00:00 del 1 de enero de 1970 GMT. Por ejemplo la utilísima función `date()`, que recibe un string con el formato que se desea para la fecha y un timestamp de Unix para introducir la fecha que se pretende formatear.

El problema que se encontrará con los cálculos de fecha es que el timestamp empieza a contar desde 1970. ¿Qué pasa con las fechas anteriores? Además, el timestamp, como estructura de datos, tiene un tamaño fijo, es decir, una capacidad limitada hasta el año 2038. En concreto soporta los siguientes intervalos de años:

Windows: desde 1970 hasta 2038
Unix: desde 1901 hasta 2038

Este artículo pretende dar a conocer un mecanismo para poder trabajar con fechas fuera de este intervalo.

Librería `adodb_date_time_library`

Existen varios métodos de solventar este problema, pero nosotros vamos a hablar de uno muy sencillo y fácil de utilizar. Se trata de la librería `adodb_date_time`, que se puede descargar y utilizar gratuitamente en nuestras aplicaciones.

Esta librería soporta fechas desde el año 100 D.C. hasta billones de años en el futuro.

Se puede descargar desde http://php.weblogs.com/adodb_date_time_library

Para utilizarla, simplemente debemos sustituir algunas de las funciones típicas de fechas

de PHP por las que implementa la librería.

getdate() reemplazar por adodb_getdate()

date() reemplazar por adodb_date()

gmdate() reemplazar por adodb_gmdate()

mktime() reemplazar por adodb_mktime()

gmmktime() reemplazar por adodb_gmmktime()

Nosotros ya hemos probado la librería con éxito en alguna creación nuestra, así que la recomendamos encarecidamente cuando supongamos que las fechas con las que vamos a trabajar se salen del intervalo del timestamp Unix.

Para ampliar esta información será imprescindible acceder a la página de inicio de la librería y enterarnos de todos los detalles sobre su uso.

http://php.weblogs.com/adodb_date_time_library

Informe de **Miguel Angel Alvarez**

Director DesarrolloWeb.com

Mail: miguel@desarrolloweb.com

Listas de elementos con colores alternos en PHP

Vamos a ver como crear una lista de elementos en una tabla creada por filas con colores alternos. Es decir, vamos a mostrar una lista de elementos mostrados en un conjunto de filas que pueden tener uno de dos colores y que se van alternado, primero uno y luego el otro. Para entender mejor el ejercicio sería interesante [ver el efecto buscado en una página aparte](#).

Esta disposición en filas de distintos colores puede dar un aspecto un poco más completo a nuestro diseño y servir de guía visual para que la tabla se entienda con un golpe de vista y se conozca mejor cuales son los datos que pertenecen a cada fila.

Para conseguir los colores alternos vamos a utilizar una variable que llevará la cuenta del número de filas y si la fila es impar mostrar un color distinto que el de la fila par. Para deducir si una fila es par o impar podemos dividir el número de fila entre 2 y si el resto de esa división es igual a cero, es que el número es par y si es distinto de cero, entonces el número es impar.

El operador %

Para obtener el resto de una división utilizamos el operador %, llamado operador de módulo. Insistimos: % devuelve el resto de una división entre los dos valores que se apliquen a la operación. Por ejemplo $2 \% 2$ valdrá 0, y la operación $6 \% 5$ dará 1.

Bucle con filas de colores alternos

Veamos ahora el código de un bucle en el que se utilicen filas con colores alternos. Utilizaremos la [tabla del ejercicio del libro de visitas](#), relatado en un [Manual práctico](#) de DesarrolloWeb.com.

El recorrido será parecido a otros que hemos realizado por los elementos del resultado de

una consulta SQL contra una base de datos. La única tarea a realizar que no hayamos visto con anterioridad en los artículos de PHP publicados en este sitio, consiste en que se lleva una cuenta del número de filas y por cada fila se comprueba si el módulo del número de fila y el número 2 es cero o no. Si es cero se colocará un color y si no vale cero se colocará otro color, con lo que se creará esa alternancia.

El código es el siguiente.

```
<?
//creo la sentencia sql para atacar a la base de datos.
$ssql = "SELECT * FROM librovisitas_php";
$ssql .= " ORDER BY id_librovisitas_php limit 10";

//ejecuto la sentencia para extraer un conjunto de resultados
$resultid = mysql_query($ssql,$conn);

//coloco la cabecera de la tabla
?>
<table width=500 align=center>
<tr bgcolor="bbbbbb" align=center>
    <td><b>Nombre</b></td>
    <td><b>Email</b></td>
    <td><b>Valoracion</b></td>
</tr>
<?
//creo e inicializo la variable para contar el número de filas
$num_fila = 0;

//bucle para mostrar los resultados
while ($damefila=mysql_fetch_object($resultid)){
    echo "<tr ";
    if ($num_fila%2==0)
        echo "bgcolor=#ddddd"; //si el resto de la división es 0 pongo un color
    else
        echo "bgcolor=#dddddff"; //si el resto de la división NO es 0 pongo otro color
    echo ">";
?>
    <td><?echo $damefila->nombre;?></td>
    <td><?echo $damefila->email;?></td>
    <td><?echo $damefila->valoracion;?></td>
</tr>
<?
    //aumentamos en uno el número de filas
    $num_fila++;
} //cierro el while
?>
</table>
```

Se puede [ver el resultado de ejecutar este script en una página aparte](#).

Informe de **Miguel Angel Alvarez**
Director DesarrolloWeb.com
Mail: miguel@desarrolloweb.com

Template Power

Una de las claves de un desarrollo en la web limpio y fácil de mantener es separar el código de la presentación. En esa línea de ideas, las tecnologías modernas, como XML o CSS, ya están pensadas para hacer posible la separación de forma y contenido, cosa que

resulta impensable con HTML.

El caso que nos ocupa, Template Power, consiste en un módulo que viene a hacer posible la separación del código PHP y el código HTML de una manera fácil y rápida. Dicho de otra manera, Template Power permite escribir el código HTML en unos archivos llamados plantillas y el código PHP en páginas que no tendrían necesidad de incluir una sola etiqueta HTML, sino que solamente llamarían a las distintas plantillas necesarias para hacer la página.

El producto es gratuito para uso personal y se puede utilizar en versiones de PHP 4.0.1 o superior. Su página de inicio, donde se puede descargar y acceder al manual es <http://templatepower.codocad.com>.

Método de trabajo

Vamos a describir aquí el método de trabajo usando Template Power, aunque no deseamos alargarnos mucho, ya que el propio producto tiene un completo manual que podemos encontrar incluso en castellano.

Un primer paso sería crear una plantilla, que contendrá código HTML, junto con algunas marcas propias del sistema Power Template. Entre estas podremos encontrar variables, que permiten sustituir el nombre de una variable por un valor, y los bloques, que permiten incluir otras plantillas o una lista de elementos con distintos datos.

La sintaxis de una plantilla sería algo parecido a esto.

```
<div align=center>{ nombrecategoria} </div>
<hr align="center">
<table width="300" cellpadding="2" cellspacing="2" align=center>
<!-- START BLOCK : productos-->
<tr><td>
<b><a href="ficha_producto.php?id={ idproducto}">{ nombreproducto} </a></b><br>
{ descripcionproducto}
</td></tr>
<!-- END BLOCK : productos -->
</table>
```

En la plantilla del ejemplo podríamos estar visualizando una categoría de productos de una tienda virtual. En la categoría de productos habría un nombre y una lista de productos asociados a la categoría, que podrían ser uno o varios.

Las variables, serían las introducidas entre llaves, por ejemplo {nombrecategoria} y se podrían sustituir por un dato.

Los bloques estarían definidos por una línea de código como la que sigue:

```
<!-- START BLOCK : productos-->
```

Con su correspondiente línea de cierre de bloque, situada un poco más abajo. En este caso el bloque sirve para incluir una repetición de productos dentro de una categoría.

Como hemos podido comprobar, en las plantillas no se incluye ni una sola línea PHP. Ahora veremos cómo se programarían las páginas PHP para hacer uso de una plantilla como la que hemos visto.

<?

```
require "../clases/TemplatePower.inc.php";

$t = new TemplatePower("plantillas/cat_productos.tpl");
$t->prepare();

$t->assign(array(
    nombrecategoria => 'Material de cocina',
));

$sql = "select * from productos where nombrecategoria='Material de cocina'"
$productos = mysql_query($sql);
while ($fila=mysql_fetch_object($productos)){
    $t->newBlock("productos");
    $t->assign(array(
        nombreproducto => $fila->nombreproducto,
        idproducto => $fila->idproducto,
        descripcionproducto => $fila->decripproducto
    ));
}
$t->printToScreen();
?>
```

Nota: Estos scripts están pensados únicamente para ilustrar el funcionamiento de Template Power. Han sido extractados de una aplicación que he realizado recientemente con este sistema, por lo que pueden estar incompletos o incorrectos sintacticamente.

El uso de plantillas desde el código PHP se realiza a través de una clase llamada TemplatePower, que implementa una serie de métodos útiles para realizar los trabajos necesarios con la plantilla de una manera rápida.

Lo primero es incluir el código de la clase, que se puede descargar desde el sitio de TemplatePower de manera gratuita. Más tarde ya estamos en condiciones de instanciar el objeto plantilla, con el constructor de la clase, indicando por parámetro el archivo desde donde se va a tomar el código HTML.

El método prepare() sirve para que se analice la plantilla. Después de ejecutarlo ya podremos asignar valores a las variables o bloques dentro de las plantillas.

Con el método assign, que recibe un array con los pares Nombre de variable => Valor de la variable, podemos asignar un valor a cada una de las variables de la plantilla.

En el código PHP podremos hacer consultas a la base de datos para extraer los valores e las variables, como es el caso del ejemplo, en el que se extraen todos los productos de una categoría. Con un bucle while, que recorre todos los productos de la base de datos, podemos crear un bloque para cada uno de los productos obtenidos. Para ello está el método newBlock(), que recibe el nombre del bloque que se debe generar. Dentro de cada bloque podemos asignar las variables que contiene la plantilla dentro de dicho bloque, también con el método assign().

Finalmente, cuando se han asignado todas las variables y bloques a la plantilla, se debe invocar el método printToScreen() para escribir el contenido generado por la plantilla y los valores dentro de la página web.

Otras posibilidades

La librería Template Power ha evolucionado bastante con el tiempo. Están por la versión 3.0, que soporta casi cualquier cosa que podamos necesitar en el trabajo de separar el

código HTML del PHP.

Incluye soporte para plantillas anidadas, posibilidad de incluir códigos PHP, que se ejecutarían también. Asignación global de variables y asignación en una única sentencia de varias variables.

También podremos salvar en disco y utilizar una plantilla ya analizada, incluso guardar plantillas en la base de datos.

Conclusión

Template Power es un producto muy interesante para separar el código PHP del código HTML. Como es gratuito lo podemos utilizar o probar sin problemas ni desembolsos, aunque si deseamos utilizarlo para un sistema comercial estamos obligados a pagar una tasa, que es realmente asequible (3 US\$).

En definitiva, es un sistema que puede subir mucho la calidad de nuestros códigos e incluso aumentar nuestra productividad. Para trabajar con Template Power necesitaremos acostumbrarnos un poco al mecanismo planteado y conocer bien las posibilidades del producto. Todo ello será muy fácil porque en la página de inicio de Template Power (<http://templatepower.codocad.com>) tenemos unos buenos ejemplos y un manual en varias lenguas.

Informe de **Miguel Angel Alvarez**
Director DesarrolloWeb.com
Mail: miguel@desarrolloweb.com

Convertir fechas entre MySQL y castellano, en PHP

Las fechas son uno de esos típicos asuntos que pueden hacer que nos rompamos la cabeza a la hora de programar una página. Razón de ello es que tienen distintos formatos dependiendo del país, del lenguaje de programación o de la base de datos que estemos utilizando.

Cuando utilizamos la tecnología PHP solemos trabajar con la base de datos MySQL. En estos dos sistemas los formatos de fechas cambian sensiblemente, así que será muy interesante conocer una manera rápida de pasar de un formato de fecha a otro, dependiendo de dónde vamos a utilizar esa fecha. Pues, si trabajamos con MySQL deberemos expresar la fecha de una manera distinta a la que lo haríamos a la hora de mostrarla en la página para que la entienda fácilmente un lector hispano.

En muchos casos, debemos vérnoslas entre dos tipos de formatos distintos, aunque podría ser peor. Por ejemplo, si la página estuviese en varios idiomas, sería importante escribir correctamente las fechas en cada uno de los idiomas.

Dejando temas relacionados con el idioma aparte -concentrándonos tan sólo en el Español-, en nuestras páginas programadas en PHP y con base de datos MySQL, tendremos que trabajar con dos formatos. Por un lado tenemos las fechas en castellano, que tienen el formato dd/mm/aaaa y por otro lado tenemos el formato de MySQL, que tiene la sintaxis aaaa-mm-dd.

Lo más cómodo, tal como vemos nosotros este problema, es crear un par de funciones

que conviertan las fechas de un formato a otro. Habrá una función que convertirá la fecha de MySQL a Castellano y otra que lo convierta de Castellano a MySQL.

```
////////////////////////////////////
//Convierte fecha de mysql a normal
////////////////////////////////////
function cambiaf_a_normal($fecha){
    ereg( "([0-9]{2,4})-([0-9]{1,2})-([0-9]{1,2})", $fecha, $mifecha);
    $lafecha=$mifecha[3]."/".$mifecha[2]."/".$mifecha[1];
    return $lafecha;
}

////////////////////////////////////
//Convierte fecha de normal a mysql
////////////////////////////////////

function cambiaf_a_mysql($fecha){
    ereg( "([0-9]{1,2})/([0-9]{1,2})/([0-9]{2,4})", $fecha, $mifecha);
    $lafecha=$mifecha[3]."-".$mifecha[2]."-".$mifecha[1];
    return $lafecha;
}
```

Las funciones utilizan expresiones regulares que no hemos visto todavía, así que no vamos a tratar de explicar cómo funcionan, sino que explicaremos cómo utilizarlas.

Mostrar en la página una fecha en castellano

Si tenemos una fecha en formato MySQL y deseamos colocarla en una página haremos algo como sigue.

Suponemos que la fecha está extrayéndose a través de una consulta a la base de datos y la tenemos en una variable llamada \$fila->fecha. Además, colocamos la fecha en un campo de formulario.

```
<input type="text" name="fecha" value="<?echo cambiaf_a_normal($fila->fecha);?>">
```

Colocar en la base de datos una fecha en formato MySQL

Cuando el usuario nos manda una fecha, por ejemplo, a través de un formulario con un campo como el que acabamos de ver, lógicamente, escribirá la fecha en castellano. Pero nosotros deseamos guardarla en una base de datos en un formato distinto, así que habremos de convertirla.

Suponemos que tenemos la fecha en una variable llamada \$fecha y que está en formato castellano. Además, queremos colocarla en una sentencia SQL que deseamos ejecutar en la base de datos para insertar un registro que contiene, entre otros datos, la fecha que el usuario ha escrito.

```
mysql_query ("insert into documento (titulo_documento, fecha_documento, cuerpo_documento) values ('$titulo_documento', '' . cambiaf_a_mysql($fecha) . '' , '$cuerpo_documento')");
```

Informe de **Miguel Angel Alvarez**
Director DesarrolloWeb.com
Mail: miguel@desarrolloweb.com

Paginador PHP usando pear y templates

A continuación podemos ver un script para paginar datos utilizando pear y templates bastante fácil de entender. Lo he documentado todo lo que he podido en el propio código del script.

El uso del script es totalmente libre y gratuito. Espero que les sirva.

El script combina el uso de templates y la paginación de datos utilizando pear. Para que funcione deberás asegurarte de tener las siguientes librerías:

```
require_once('DB.php');  
require_once('DB/Pager.php');  
require_once('HTML/ITX.php');
```

Referencia: Tenemos más scripts para paginar resultados en PHP, englobados dentro del [Taller de PHP](#):

[Paginación de resultados con PHP y MySQL](#)

[Paginación PHP y MySQL. Ejemplo 2.](#)

Para probar el script necesitas tener una carpeta Templates con el archivo template.html que esta en el archivo .zip que puedes [descargar en este enlace](#).

La estructura de la tabla para que corra este script es:

Nombre de la Tabla: "Invitaciones"

Campos:

ID_Invitacion
NombreInvitado
email

No se os olvide colocar el nombre de la base de datos donde dice "nombre_de_la_base".

Si encuentras mejoras al script, por favor infórmanos en diems2@yahoo.com.ar

Puedes [descargar el script aquí](#).

Informe de **Diego Villar**

Mail: diems2@yahoo.com.ar

Contador mejorado para páginas PHP

Dadas mis necesidades he tenido que modificar el script del [contador simple para páginas PHP](#), y me gustaría compartirlo con todos vosotros. La diferencia está en que ahora almacena tres datos en el archivo que hace de contador:

- 0) el número de mes de la última visita
- 1) el número de visitas del mes
- 2) el número de visitas totales

El script queda como sigue:

```
function interface_contador(){
```

```

$archivo = "contador.txt";
$info = array();

//comprobar si existe el archivo
if (file_exists($archivo)){
    // abrir archivo de texto y introducir los datos en el array $info
    $fp = fopen($archivo,"r");
    $contador = fgets($fp, 26);
    $info = explode(" ", $contador);
    fclose($fp);

    // poner nombre a cada dato
    $mes_actual = date("m");
    $mes_ultimo = $info[0];
    $visitas_mes = $info[1];
    $visitas_totales = $info[2];
}else{
    // inicializar valores
    $mes_actual = date("m");
    $mes_ultimo = "0";
    $visitas_mes = 0;
    $visitas_totales = 0;
}

// incrementar las visitas del mes según si estamos en el mismo
// mes o no que el de la ultima visita, o ponerlas a cero
if ($mes_actual==$mes_ultimo){
    $visitas_mes++;
}else{
    $visitas_mes=1;
}
$visitas_totales++;

// reconstruir el array con los nuevos valores
$info[0] = $mes_actual;
$info[1] = $visitas_mes;
$info[2] = $visitas_totales;

// grabar los valores en el archivo de nuevo
$info_nueva = implode(" ", $info);
$fp = fopen($archivo,"w+");
fwrite($fp, $info_nueva, 26);
fclose($fp);

// devolver el array
return $info;
}

```

Este código devuelve un array cuando es llamado, con 3 elementos (el mes actual, visitas del mes, visitas totales) que cada uno puede utilizar como quiera.

Por cierto, me olvidaba decir que he añadido una cláusula condicional que verifica si existe el archivo \$contador en el directorio de la página, previamente a ser leído. Lo he hecho para evitar que diera error la primera vez que se ejecutaba el script. De esta forma, ya no hace falta preocuparse por poner en el servidor un archivo \$contador a cero antes de ejecutar el código por primera vez.

Se puede [ver el ejemplo en marcha en esta página](#). También se puede [descargar el código de la página del ejemplo](#), con la función y una sencilla muestra de cómo utilizarla.

Upload de archivos con PHP

En PHP tenemos muchas funcionalidades desarrolladas desde el principio y sin necesidad de instalar ningún añadido en nuestro servidor. Es el caso de subir archivos a un servidor web por HTTP y a través de una página con un formulario, donde se permite seleccionar el archivo que queremos cargar de nuestro disco duro.

El ejemplo se encuentra bien documentado en un montón de páginas para desarrolladores, sin ir más lejos en la página de la propia tecnología: <http://www.php.net/manual/es/features.file-upload.php>. Nosotros en este caso vamos a intentar ir un poco más allá, realizando un par de comprobaciones al subir el fichero y combinando en el mismo formulario campos de tipo file y tipo text.

El formulario para subir seleccionar los archivos

Es un formulario cualquiera, pero tiene una serie de particularidades y campos file, que no solemos utilizar habitualmente.

```
<form action="subearchivo.php" method="post" enctype="multipart/form-data">
  <b>Campo de tipo texto: </b>
  <br>
  <input type="text" name="cadenatexto" size="20" maxlength="100">
  <input type="hidden" name="MAX_FILE_SIZE" value="100000">
  <br>
  <br>
  <b>Enviar un nuevo archivo: </b>
  <br>
  <input name="userfile" type="file">
  <br>
  <input type="submit" value="Enviar">
</form>
```

Para empezar vemos que se ha colocado un nuevo atributo en el formulario: `enctype="multipart/form-data"`, necesario para subir en un mismo formulario datos y archivos.

También tenemos el campo hidden `MAX_FILE_SIZE`, que sirve para indicar el tamaño en bytes de los archivos a subir. Este campo algunos navegadores no tienen porque entenderlo o hacerle caso. Además, es fácil saltarse esa protección, por lo que deberemos en las propias páginas PHP comprobar que el archivo tenga el tamaño que deseamos.

Por último, tenemos el campo tipo file, donde se seleccionará el archivo a subir. También hemos colocado un campo de tipo text, para subir datos por POST de tipo texto acompañados a los datos binarios del archivo.

Página que sube los archivos

Esta página debe hacer las comprobaciones necesarias para saber si las características del archivo a subir son las que deseamos y realizar la copia del archivo en un directorio del servidor.

Para hacer las comprobaciones, PHP nos crea una serie de variables que podemos acceder con la información del archivo enviado.

`$HTTP_POST_FILES['userfile']['name']`
El nombre original del fichero en la máquina cliente.

`$HTTP_POST_FILES['userfile']['type']`
El tipo mime del fichero (si el navegador lo proporciona). Un ejemplo podría ser "image/gif".

`$HTTP_POST_FILES['userfile']['size']`
El tamaño en bytes del fichero recibido.

`$HTTP_POST_FILES['userfile']['tmp_name']`
El nombre del fichero temporal que se utiliza para almacenar en el servidor el archivo recibido.

```
<?
//tomo el valor de un elemento de tipo texto del formulario
$cadenatexto = $_POST["cadenatexto"];
echo "Escribió en el campo de texto: " . $cadenatexto . "<br><br>";

//datos del arhivo
$nombre_archivo = $HTTP_POST_FILES['userfile']['name'];
$tipo_archivo = $HTTP_POST_FILES['userfile']['type'];
$tamano_archivo = $HTTP_POST_FILES['userfile']['size'];
//compruebo si las características del archivo son las que deseo
if (!((strpos($tipo_archivo, "gif") || strpos($tipo_archivo, "jpeg")) && ($tamano_archivo < 100000))) {
    echo "La extensión o el tamaño de los archivos no es correcta. <br><br><table><tr><td><li>Se permiten
archivos .gif o .jpg<br><li>se permiten archivos de 100 Kb máximo.</td></tr></table>";
}else{
    if (move_uploaded_file($HTTP_POST_FILES['userfile']['tmp_name'], $nombre_archivo)){
        echo "El archivo ha sido cargado correctamente.";
    }else{
        echo "Ocurrió algún error al subir el fichero. No pudo guardarse.";
    }
}
?>
```

Para empezar, recogemos el campo de texto enviado por POST, de la forma habitual. Aunque esto no tenga nada que ver con subir archivos, es muy normal que en el mismo formulario deseemos mezclar varios tipos de información.

Luego se recogen los datos necesarios del archivo, como su nombre, extensión y tamaño para, en el siguiente if, comprobar que la extensión sea .gif o .jpg y que el tamaño menor que 100000 bytes.

Si el archivo tenía las características deseadas, se puede subir al servidor. Para ello se utiliza la función `move_uploaded_file()`, que recibe el nombre del archivo temporal que se desea subir y el nombre del archivo que se desea dar.

Cuando se sube el archivo, el servidor lo copia en una localización temporal para que seamos nosotros los que elijamos la posición definitiva donde queremos que se almacene. Si no lo copiamos a ningún sitio, después de la ejecución de la página, se borra de su localización temporal.

La función `move_uploaded_file()` se utiliza para mover el archivo a la posición definitiva. Recibe por un lado el nombre temporal del fichero y por otro el nombre que deseamos colocarle definitivamente y, si se desea, la ruta para llegar al directorio donde queremos guardarlo. En el caso del ejemplo sólo se indica el nombre del archivo, por ello el fichero

se subirá al mismo directorio donde están las páginas PHP que hacen el upload. Esta función devuelve un booleano que indica si hubo o no éxito al subir el archivo.

Nota: Es importante señalar que el upload de archivos es un proceso crítico que puede dar lugar a errores y agujeros de seguridad. Por ejemplo, si los directorios destino están protegidos contra escritura, nos dará un error. Podemos ver los [errores comunes relatados en la página de PHP](#).

Recomendamos una vez más ampliar esta información en la página de PHP: <http://www.php.net/manual/es/features.file-upload.php>

Se pueden [descargar los códigos del ejemplo en este enlace](#).

Informe de **Miguel Angel Alvarez**
Director DesarrolloWeb.com
Mail: miguel@desarrolloweb.com

Bucle para recibir todas las variables por POST en PHP

Vamos a ver una manera muy rápida de recibir todas las variables de un formulario, enviado por post, en el lenguaje PHP. Os aseguro que es una pequeña porción de código que os ahorrará escribir un montón de líneas de código.

Quién no se ha visto alguna vez en la tediosa tarea de recibir un montón de datos de un formulario, asignando una por una todas las variables en PHP? Eso se hacía con líneas como ésta:

```
$nombre = $_POST["nombre"];  
$edad = $_POST["edad"];  
$ciudad = $_POST["ciudad"];  
....
```

Si el formulario tuviera 10 elementos no sería muy pesado escribir las 10 líneas de código, pero si fueran 50 o 100 la cosa sería mucho menos agradable. El código que vamos a ver ahora nos solucionará la vida en esos casos.

```
foreach($_POST as $nombre_campo => $valor){  
    $asignacion = "\$" . $ nombre_campo . "=" . $valor . " ";  
    eval($asignacion);  
}
```

Se realiza un bucle foreach que va recorriendo cada uno de los elementos del post. En cada iteración, se van accediendo a todos los elementos del post y se guarda en \$ nombre_campo el nombre del campo recibido por el formulario y en \$valor, el valor que se había introducido en el formulario.

Todo lo anterior se deduce de la primera línea. En las siguientes se compone en cada iteración, cada una de las asignaciones que deberíamos haber escrito manualmente. Es decir, en la variable asignación guardaremos una línea de código PHP que realiza la declaración de la variable de formulario dentro de PHP y su inicialización con el valor que se hubiera escrito.

En la siguiente línea, donde está la función eval(), se ejecuta la sentencia generada en el anterior paso. La función eval() de PHP ejecuta el contenido de una cadena de caracteres

como si fuera una sentencia PHP. (Podemos ver la documentación de la función eval() en la página de PHP <http://es.php.net/manual/es/function.eval.php>)

Esperamos que os haya interesado este minúsculo, pero útil, código PHP.

Informe de **Miguel Angel Alvarez**
Director DesarrolloWeb.com
Mail: miguel@desarrolloweb.com

Contador PHP con imágenes

Aquí les dejo el código PHP de un contador mejorado con la inclusión de imágenes para generar el número de visitas en lugar de utilizar texto como se venía realizando en el artículo [contador simple de páginas PHP](#).

Para poner en marcha el ejemplo sólo hay que crear los números del 0 al 9 en formato imagen. Es preferible que los números sean menores a estas propiedades: width="17" height="28"

Otra cosa, parte del código fue tomado de [este contador](#). Lo que he incluido yo es lo que hace que llame a las imágenes, que deben ser nombradas "0.gif", "1.gif", ..., "9.gif".

```
<html>
<head>
<title>Contador PHP con imágenes</title>
</head>

<body>
<?php
$archivo = "contador.txt";
$contador = 0;

$fp = fopen($archivo,"r");
$contador = fgets($fp, 26);
fclose($fp);

++$contador;

$fp = fopen($archivo,"w+");
fwrite($fp, $contador, 26);
fclose($fp);
?>

<table width="102" border="0" align="center" cellpadding="0" cellspacing="0">
<tr>
<?php

/* Arreglo de 0-9 nombre de los archivos gifs*/
$numero[0]="0.gif";
$numero[1]="1.gif";
$numero[2]="2.gif";
$numero[3]="3.gif";
$numero[4]="4.gif";
$numero[5]="5.gif";
$numero[6]="6.gif";
$numero[7]="7.gif";
$numero[8]="8.gif";
$numero[9]="9.gif";
```

```
/*Se crea variable para que contenga la longitud de la cadena*/
/*es a partir de ahí donde se sabe que mostrará el contador en GIFS*/

$longitud = strlen ($contador);

/* Bucles para mostrar los números*/
$hasta = 6-$longitud;

For ($celda = 1;$celda <= $hasta;$celda++)
{
    echo "<td width=\"17\" height=\"28\" valign=\"top\"><div align=\"center\"><img src=\""$numero[0]\"></div></td>";
}
$hasta = $longitud-1;
For ($celda = 0;$celda <= $hasta;$celda++)
{
    $num = substr ($contador, $celda, 1);
    echo "<td width=\"17\" height=\"28\" valign=\"top\"><div align=\"center\"><img src=\""$numero[$num]\"></div></td>";
}

?>

</tr>
</table>
</div>
</body>
</html>
```

Informe de **Josue Gutiérrez Olivares**
Mail: ratasoft@hotmail.com