

Pasar las variables SESSION, POST Y GET a variables normales en PHP

Hace un par de semanas se me encomendó migrar un sitio (desarrollado en php) de servidor, el tema iba relativamente en paz hasta que di cuenta de un problema recurrente en este tipo de situaciones, el servidor antiguo tenia las variables globales en On y el nuevo las tenia en Off ... recurrente no ?

La verdad es que no quise complicarme la vida como otras veces, y me di cuenta de algo que me podría ahorrar un par de horas de trabajo, todos los php hacían un include a un archivo php, recurrente también no?

Así es que me puse a pensar en un pedazo de código que me permitiera tomar estos 3 tipos de variables (SESSION, POST Y GET) y me las dejara como variables normales, si no lo tienen claro el ejercicio es mas o menos el siguiente :

Supongamos que la variable viene desde un formulario via POST, el código decía algo así ...

```
<?
if ($variable == "algo")
{
    echo "esta variable hace alguna cosa";
}
?>
```

Es decir, le faltaba esta instrucción antes del if

```
<?php
$variable = $_POST["variable"]; // le faltaba esta instrucción antes del if
if ($variable == "algo")
{
    echo "esta variable hace alguna cosa";
}
?>
```

Tenia 2 opciones, o me ponía a buscar TODAS estas situaciones y las arreglaba una a una, o colocaba un pedazo de código en archivo al que todos los demás le hacían un include que me corrigiera este "error" propio de aquellos que estamos a trabajar con los "servidores en producción", o sea, con las variables globales en On.

Bueno, después de muchos cabecearme encontré la solución, y aquí la muestro a todos uds.

```
<?
if($_POST)
{
    $keys_post = array_keys($_POST);
    foreach ($keys_post as $key_post)
    {
        $$key_post = $_POST[$key_post];
        error_log("variable $key_post viene desde $_POST");
    }
}
```

```
}  
  
if($_GET)  
{  
    $keys_get = array_keys($_GET);  
    foreach ($keys_get as $key_get)  
    {  
        $$key_get = $_GET[$key_get];  
        error_log("variable $key_get viene desde $_GET");  
    }  
}  
  
if($_SESSION)  
{  
    $keys_sesion = array_keys($_SESSION);  
    foreach ($keys_sesion as $key_sesion)  
    {  
        $$key_sesion = $_SESSION[$key_sesion];  
        error_log("variable $key_sesion viene desde $_SESSION");  
    }  
}  
  
?>
```

Son básicamente 3 if que hacen lo mismo, toman las claves del arreglo en cuestión (SESSION , POST o GET) y generan una "variable variable", creo que hay un artículo en desarrolloweb que habla de este tema, y esto hace toda la magia, el error_log fue una implementación para ver que variables se iban ocupando (si no tienes acceso al servidor puedes sacarlo sin problemas).

Para alguien que guste de las funciones puede también guardarlo como función y llamarla cuando lo necesite. O también iterar el proceso con las 3 variables en cuestión para no tener que hacer 3 if, o pasarle cualquier arreglo para que haga el mismo proceso ... en fin, la idea esta, solo espero haber contribuido en algo al ahorro de tiempo de alguno de ustedes.

Artículo por [Juan Edgardo Jorquera Uribe](#)

Bucle para recibir todos los datos de una fila de un recordset con PHP

Vamos a realizar una función para recibir todos los datos de una fila de un recordset (conjunto de registros extraídos de una base de datos, Mysql en el caso de este artículo) y declararlos como variables globales a la página. Se trata de una función que realiza un recorrido genérico a todos los datos de cualquier fila y cualquier recordset y va generando las variables globales para trabajar con esos datos.

Las variables globales tendrían como nombre el mismo del campo (que es el mismo nombre que la columna de la tabla asociada a ese dato) y como valor, el dato que guarda esa fila en la columna determinada.

Nota: Este artículo es similar a otro en el que se realiza un [recorrido genérico a las variables recibidas por POST](#). Las explicaciones de ese artículo pueden venir bien para comprender mejor este.

Para una tabla como esta:

id_clie	nombre	telefono_
---------	--------	-----------

nfe	_cliente	cliente
1	Pepe	90 000 00 00
2	Juan	99 888 88 88
3	María	999 99 99 99

Si la función recibiera una fila cualquiera, se generarían variables con nombres `id_cliente`, `nombre_cliente` y `telefono_cliente` y sus valores serían los de esa fila en concreto. Para la fila 1, se generarían las variables:

```
id_cliente=1
nombre_cliente="Pepe"
telefono_cliente="90 000 00 00"
```

El recorrido será genérico, por lo que servirá para cualquier tabla, con cualquier número y nombre de las columnas. Las variables generadas serán siempre las columnas que existan, con los valores de la fila que se haya recibido por parámetro.

Esta función es muy útil para recibir los datos de una fila y meterlos en variables globales, para luego operar con ellos. Como el recorrido es genérico, da igual el número, nombre y tipo de los campos que se van a meter en las variables.

```
function recibe_fila($fila){
    foreach($fila as $nombre_campo => $valor){
        if (gettype($nombre_campo)!="integer"){
            $asignacion = "\$GLOBALS[\" . $nombre_campo . "\"]=" . $valor . " ";
            eval($asignacion);
            //echo $asignacion . "<br>";
        }
    }
}
```

Un ejemplo de uso de esta función

Veamos cómo utilizar esta función para recibir los datos de una fila. Se supone que antes de ejecutar estas líneas se debe haber abierto una conexión con la base de datos. También se supone que después de ejecutar este código, se debería cerrar esa conexión con la base de datos.

```
$ssql="select * from cliente where id_cliente=2";
$rs=mysql_query($ssql);
$fila=mysql_fetch_array($rs);
recibe_fila($fila);
```

Después de estas líneas, se habrán creado las variables globales con los datos de la fila extraída de la tabla cliente, una variable para cada una de las columnas de la fila.

*Artículo por **Miguel Angel Alvarez***

Función PHP segura para recibir los datos de un formulario

La idea de este artículo es realizar una función que nos simplifique la tarea de recibir en variables los datos que nos llegan desde un formulario por el método POST, pero asegurando nuestra aplicación, para no declarar como variable aquello que no estamos esperando.

Ya hicimos una primera aproximación a este objetivo en el artículo [Bucle para recibir todas las variables por POST en PHP](#). Aunque la solución propuesta en dicho artículo no era del todo adecuada, por hacer más vulnerable nuestro script, ya que rebaja la seguridad de nuestro código como si utilizásemos `register_globals = on`.

Nota: Podemos saber qué es esto de `register_globals` y por qué afecta a la seguridad de nuestras aplicaciones PHP en la FAQ: [register_globals y seguridad en PHP](#)

El ejemplo que estamos tratando sólo recibe las variables por POST que nosotros esperamos recibir y no todas las variables que el formulario pueda contener, lo que realmente mejora la seguridad. Para ello, la función `recibe_post()` va a recibir un parámetro con los nombres de las variables que debe declarar con los datos del formulario. Como nosotros decidimos qué variables se reciben, ya no corremos el riesgo que un usuario malicioso envíe otros campos por el formulario que luego se conviertan en variables en nuestro sistema.

Veamos el código de esta función:

```
function recibe_post($datos_recibir=""){
    if ($datos_recibir==""){
        foreach($_POST as $nombre_campo => $valor){
            $asignacion = "\$GLOBALS[\" . $nombre_campo . "\"]=" . $valor . ";";
            eval($asignacion);
        }
    }else{
        //es que recibo por parámetro la lista de campos que deseo recibir
        $campos = explode(",", $datos_recibir);
        foreach($campos as $nombre_campo){
            $asignacion = "\$GLOBALS[\" . $nombre_campo . "\"]=" . $_POST[\" . $nombre_campo . "\"];";
            eval($asignacion);
        }
    }
}
```

Esta función recibe una variable llamada `$datos_recibir`, que contiene los datos que se desean recibir. Ese dato es opcional, ya que si no recibe nada, se entiende que `$datos_recibir` vale "".

Lo primero que se hace es evaluar si `$datos_recibir` es un string vacío. Si es así, es que no se ha indicado qué es lo que se quiere recibir. Entonces, se recibe por formulario todo lo que haya. (Nótese que si no enviamos nada en ese parámetro se estarán recibiendo todos los datos que haya en el formulario y nuestra aplicación verá mermada la seguridad).

Ahora bien, si se recibe algo en `$datos_recibir`, es que sabemos qué datos se desean declarar como variables. En ese caso, estaremos recibiendo un string con todos los nombres de los campos del formulario que deseamos recibir, separados por comas. Imaginemos que tenemos un formulario con dos campos, uno llamado "nombre_usuario" y otro "edad_usuario". Entonces, a esta función tenemos que pasarle esos dos nombres de campos separados por comas "nombre_usuario,edad_usuario".

Para realizar las tareas de declaración de las variables, se crea un array con todos los campos

recibidos por parámetro. Para ello se utiliza la función `explode()`, que recibe un separador y un string y devuelve un array de strings, donde cada cadena es un substring del string recibido por parámetro, acotados por el separador indicado. La función `explode` se explica mejor con un ejemplo:

La función recibe dos parámetros: `explode($separador, $cadena)`. Si la llamásemos así:

```
explode("|", "pepe|juan|luis")
```

Nos generaría un array donde el primer campo (índice 0) tendría el substring "pepe", el segundo campo sería "juan" y el tercero "luis".

Una vez disponemos del array con todos los nombres de los campos que se desean recibir, se hace un recorrido de ese array para obtener cada uno de sus valores y se declaran como variables los datos que contiene el formulario para cada uno de los campos a recibir.

Para ello, primero se crea un string con el código PHP necesario para declarar esa variable, utilizando el array `$GLOBALS`, para asegurarnos que la variable se crea global a la página en lugar de local a la función. Luego se utiliza `eval()` para ejecutar el código generado para declarar la variable.

Hay más explicaciones sobre este último paso en el artículo precedente: [Bucle para recibir todas las variables por POST en PHP](#)

Con esto hemos terminado este código, que se puede utilizar para recibir sin mucho esfuerzo formularios muy grandes. Lo malo es que si cambiamos o añadimos un campo en el formulario, también tenemos que cambiar el código de llamada a esta función, para que se entere bien sobre las variables que tiene que recibir. La parte buena, es que la función no afecta a la seguridad de nuestras aplicaciones, ya que somos nosotros quienes deciden qué variables se esperan del formulario.

*Artículo por **Miguel Angel Alvarez***

Páginas multi-idioma con PHP

Para realizar una página web multi-idioma necesitamos solucionar varios problemas o casuísticas. Vamos a numerarlas rápidamente y posteriormente daremos una posible solución para cada una.

1. Traducción de textos planos
2. Traducción de textos que están insertados en una base de datos
3. Traducción de textos mezclados con valores de variables

En líneas generales, la solución pasa por tener guardados en variables todos los textos que se van a mostrar en la página. Podemos utilizar variables tal cual o bien generar un array con todos los textos a traducir, lo que puede mejorar la organización del código.

Veamos las explicaciones de los tres casos señalados anteriormente.

1) Podemos tener textos planos que traducir, es decir, textos que simplemente debemos colocarlos en un idioma u otro, dependiendo de la preferencia del usuario.

Por ejemplo, tenemos que poner escribir palabra "nombre" en la página. En español escribiremos "nombre", pero cuando se visite el sitio en inglés, escribiríamos "name".

Esto lo estamos solucionando con un fichero de texto, en el que tenemos como variables todas las palabras o frases planas que se necesitan escribir en la página, en varios idiomas. Así tenemos un fichero con las palabras y frases en español, otro con las del idioma inglés y, por ejemplo, otro con las de portugués.

Las variables que estamos utilizando son del estilo \$idioma_loquesea. Por ejemplo, en el fichero en español podemos tener varias palabras y frases como estas:

```
$idioma_nombre = "nombre";  
$idioma_direccion = "dirección";  
$idioma_error_usuario = "Hemos detectado un error con el usuario";
```

En el idioma inglés tendríamos un fichero parecido a este: (perdonar si mis traducciones no son del todo correctas)

```
$idioma_nombre = "name";  
$idioma_direccion = "address";  
$idioma_error_usuario = "We have detected an user error";
```

En las páginas multi-idioma, detectaríamos el idioma que ha seleccionado el usuario, para incluir un fichero de idioma u otro.

Luego, al mostrar un texto, podríamos sacar algo como esto:

```
echo $idioma_nombre . ": pepe";  
echo $idioma_direccion . ": C/ corona, 2";
```

Dependiendo del fichero de idioma que se haya incluido tendremos un resultado distinto. En español saldría:

```
nombre: pepe  
dirección: C/ corona, 2
```

Si hubiéramos incluido el fichero de idioma inglés, obtendríamos como salida

```
name: pepe  
address: C/ corona, 2
```

2) Podemos tener otro caso de elementos a traducir más complejo. Supongamos que tenemos una tabla de países. Los países se llaman de manera distinta en cada idioma, así que de alguna manera tenemos que almacenar el nombre del país para muchos idiomas distintos.

Esto se puede hacer de varias maneras. Por ejemplo, podríamos tener una tabla con los identificadores de los países y la traducción para cada idioma. Luego, en la página dependiendo del idioma, tendríamos que mostrar un texto u otro para el país, seleccionando la traducción que necesitamos para el país.

Por ejemplo, podríamos tener la tabla pais, de esta manera:

```
id_pais - nombre_pais_es - nombre_pais_en
1 - España - Spain
2 - Italia - Italy
3 - Francia - France
```

Luego, al recuperar los nombres de los países, podríamos hacer algo como esto:

```
if ($lenguaje_seleccionado = "ES"){
    $ssql = "select id_pais, nombre_pais_es as 'nom_pais' from pais";
}else{
    $ssql = "select id_pais, nombre_pais_en as 'nom_pais' from pais";
}
```

Luego, recuperaríamos los datos de la tabla y en el campo 'nom_pais' tendremos la traducción que necesitamos.

Pero esto no nos gusta, porque no nos estamos abstrayendo del idioma del usuario para mostrar el nombre del país. Es decir, en el código de la aplicación tenemos que hacer cosas distintas para cada idioma. Lo mejor sería programar la página igual, sin tener que preguntar en ningún momento el idioma en el que estamos trabajando, así no habrá que tocar el código nunca para incorporar nuevos idiomas, ni estamos mezclando la lógica de la aplicación con la lógica de la gestión del idioma.

Una solución para mejorar esto es utilizar un fichero de texto para los nombres de los países y tener un fichero de texto para cada idioma, de manera similar a lo que habíamos comentado para el caso anterior.

Dentro de este fichero, tendremos los nombres de los países, en un archivo independiente para cada idioma. Los nombres los podemos meter en un array para facilitar su gestión, con los índices iguales al identificador utilizado en la tabla país.

Para el idioma español tendríamos:

```
$idioma_nombre_pais[1] = "España";
$idioma_nombre_pais[2] = "Italia";
$idioma_nombre_pais[3] = "Francia";
```

Para el idioma inglés, tendríamos:

```
$idioma_nombre_pais[1] = "Spain";
$idioma_nombre_pais[2] = "Italy";
$idioma_nombre_pais[3] = "France";
```

Luego, al seleccionar los distintos países de la base de datos, la sentencia SQL será la misma:

```
$ssql = "select id_pais from pais";
```

Al mostrar los nombres de países, tan sólo tenemos que acceder al array `$idioma_nombre_pais`, con el índice del país que se desea mostrar. Es decir, el código será el mismo, aunque el resultado al visualizar el nombre de país dependerá del archivo de idioma que hayamos cargado (en español o inglés). Por ejemplo, si quisiéramos mostrar un elemento de formulario `<select>` con los distintos países el código sería:

```
echo "<select name='id_pais'>";
$ssql = "select id_pais from pais";
$rs = mysql_query($ssql);
while ($fila = mysql_fetch_object($rs)){
    echo "<option value='' . $fila->id_pais . '>";
    echo $idioma_nombre_pais[$fila->id_pais];
    echo "</option>";
}
echo "</select>";
```

3) El último caso que vamos a ver en las traducciones es una mezcla entre textos planos y valores de variables. Por ejemplo, pensemos en una frase como esta:

Tienes X documentos subidos, Y abiertos

Donde X es el valor de una variable numérica, al igual que Y. Lógicamente, en lugar de la X o la Y, lo que queremos ver es el valor numérico que tengan esas variables.

Para implementar cómodamente esta parte del sistema multi-idioma, podemos utilizar la función `printf()`, que es parecida a `echo`, pero aparte de la cadena a mostrar, permite indicar otros parámetros con valores que se sustituirán en la cadena, antes de mostrarla en la página. Veamos con un ejemplo esto.

La cadena en español que queremos mostrar es:

```
$idioma_documentos_abiertos = "tienes %1u documentos subidos, %2u abiertos";
```

En inglés, este mismo mensaje quedaría:

```
$idioma_documentos_abiertos = "You have upload %1u documents, %2u open";
```

Con "%1u" en la cadena hemos especificado un parámetro y con "%2u" hemos especificado otro parámetro. Ambos se debe sustituir por un valor que también enviaremos a `printf()`. Del parámetro "%1u", la parte del "%1" hace referencia al primer parámetro y la "u" dice que es numérico, en base 10, sin decimales.

En la llamada a `printf()` debemos especificar la cadena a mostrar y los parámetros que se deben sustituir en la cadena, en nuestro ejemplo 2.

```
printf($idioma_documentos_abiertos, 6, 4);
```

Esto tendrá como salida, para la frase en español:

tienes 6 documentos subidos, 4 abiertos

*Artículo por **Miguel Angel Alvarez***

Comprobar en PHP si una cadena tiene sólo el conjunto de caracteres permitido

En algunas ocasiones tenemos que comprobar la validez de una cadena de caracteres para ver

si contiene solamente aquellos que consideramos como válidos. Por ejemplo, si tuviéramos que validar un nombre de usuario, podríamos permitir números, letras y ocasionalmente caracteres "-" o "_", pero no otro tipo de caracteres como "+", "@", "&", etc. Además, siendo un nombre de usuario, podemos tener fijados un máximo y mínimo número de caracteres.

Esta es una de las comprobaciones más simples que se pueden hacer en una cadena, pero no por ello menos importante. Veremos en este artículo una manera de realiza esa comprobación usando las funciones de string y también usando expresiones regulares de PHP.

Usando las funciones de tratamiento de string de PHP

Vamos a realizar una función que reciba un string de un nombre de usuario y compruebe si es correcto. Como hemos dicho, la comprobación tendrá dos partes, la primera para ver si la longitud de la cadena está permitida (entre 3 y 20 caracteres) y la segunda para asegurar que los caracteres utilizados están entre los permitidos.

```
function comprobar_nombre_usuario($nombre_usuario){
    //compruebo que el tamaño del string sea válido.
    if (strlen($nombre_usuario)<3 || strlen($nombre_usuario)>20){
        echo $nombre_usuario . " no es válido<br>";
        return false;
    }

    //compruebo que los caracteres sean los permitidos
    $permitidos = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789-_" ;
    for ($i=0; $i<strlen($nombre_usuario); $i++){
        if (strpos($permitidos, substr($nombre_usuario,$i,1))===false){
            echo $nombre_usuario . " no es válido<br>";
            return false;
        }
    }
    echo $nombre_usuario . " es válido<br>";
    return true;
}
```

La primera parte es muy sencilla, simplemente se mira si la longitud de la cadena es menor que 3 o mayor que 5, porque en ese caso el nombre de usuario sería incorrecto. Para la segunda comprobación, de caracteres válidos, hemos definido una variable local a la función que contiene una cadena con todos los caracteres permitidos para el nombre de usuario. Luego, para cada carácter del string que se desea comprobar, veremos si está contenido dentro de la variable local. En el momento que la cadena con el nombre de usuario tenga un carácter que no esté entre los permitidos, diremos que el nombre de usuario es incorrecto.

Si no nos salimos de la función después de todas las comprobaciones realizadas, es que el nombre de usuario es correcto.

Usando expresiones regulares de PHP

Ahora vamos a ver un método distinto para hacer la misma comprobación, en este caso utilizando expresiones regulares de PHP. Las expresiones regulares permiten comparar la cadena con un patrón, para ver si se ajusta o no a ese patrón.

Existe una función de PHP llamada `ereg()` que recibe el patrón realizado con una expresión regular y un string para ver si se ajusta al patrón. No es el objetivo explicar las expresiones regulares, aunque comentaremos cómo hemos construido el patrón para este ejemplo. La función `ereg()` devuelve la longitud de la cadena encontrada que corresponde con el patrón y

false, si no encuentra coincidencia entre la expresión regular y el string.

Veamos primero la función que hemos creado para verificar la validez del nombre de usuario:

```
funcion comprobar_nombre_usuario_expresiones_regulares($nombre_usuario){
    if (ereg("[a-zA-Z0-9\-\_]{3,20}$", $nombre_usuario)) {
        echo "El nombre de usuario $nombre_usuario es correcto<br>";
        return true;
    } else {
        echo "El nombre de usuario $nombre_usuario no es válido<br>";
        return false;
    }
}
```

Simplemente se ejecuta la función `ereg()` y se evalúa el resultado devuelto. Si sale por el caso positivo es que era correcto el nombre de usuario. Si `ereg()` devuelve un resultado que se evalúa negativamente, el usuario era incorrecto.

La dificultad de esta función está en crear la expresión regular que necesitamos para la pasarla a `ereg()`. En este caso se realizan las dos comprobaciones mencionadas (longitud del nombre de usuario y caracteres permitidos) en un único paso.

Para entender la función necesitamos conocer algunos símbolos especiales de expresiones regulares en PHP:

"^" Significa que la cadena tiene que empezar por ahí. Por ejemplo la expresión "^hola" concordaría con cualquier cadena que empezase por "hola".

[] se utilizan para indicar un carácter entre varios que se pongan entre los corchetes. Por ejemplo `^[xyz]` concordaría con cualquier cadena que comenzase por "x", "y" o "z". Dentro de los corchetes, se pueden poner intervalos de varias letras o números. Por ejemplo `[a-z]` significa cualquier letra minúscula, desde la "a" a la "z". `[0-9]` es cualquier número del 0 al 9. `[a-zA-Z0-9]` es cualquier letra, mayúscula o minúscula, y cualquier número.

Si queremos escribir en el patrón un carácter que se utiliza como un código especial de expresiones regulares, como "-" o ".", se tiene que escribir con una contrabarra delante. Por ejemplo "\-" o "\."

Luego, `[a-zA-Z0-9\-_]` es cualquier carácter de los permitidos (letras minúsculas y mayúsculas, números, "-" y "_").

{ } Las llaves se utilizan para especificar repeticiones de lo que va delante. Por ejemplo `[0-9]{4}` quiere decir que hay 4 números. `a{3,}` quiere decir por lo menos 3 repeticiones de "a". `[a-zA-Z0-9\-_]{3,20}` quiere decir que cualquiera de los caracteres permitidos en nuestros nombres de usuario, repetidos de 3 a 20 veces.

Por último, el carácter "\$" significa que la cadena tiene que terminar ahí. Por ejemplo, la expresión regular "adios\$" concordaría con cualquier cadena que terminase en "adios".

Nuestra expresión regular: `^[a-zA-Z0-9\-_]{3,20}$` significa que empiece la cadena por cualquier carácter permitido, que se encuentren de 3 a 20 caracteres permitidos y que la cadena termine después de esas repeticiones. Si el nombre de usuario cumple ese patrón, es que es correcto.

Eso es todo. Hemos visto dos formas de hacer lo mismo, aunque con expresiones regulares queda un código más elegante. Espero que también sirva de introducción para los que nunca antes habían trabajado con ellas.

Artículo por Miguel Angel Alvarez

Edición con PHP de varios registros de la base de datos de una sola vez

En el manual básico de PHP ya se explicó [como realizar una edición de los datos de una tabla de base de datos](#). En el ejemplo visto en ese artículo se podía seleccionar un registro, para editar sus datos. Pero imaginemos que necesitamos editar decenas o cientos de registros, entonces puede ser demasiado laborioso tener que editar los registros de uno en uno. En esos casos, tal vez nos interese hacer algo para que se puedan editar varios registros a la vez.

Lo que vamos a hacer es un sistema de edición que permita actualizar varios registros de la base de datos en una única acción, es decir, construir un formulario que muestre varios registros para editar, con un único botón de submit para enviar el formulario completo. La dificultad estriba en recibir del formulario todos los registros a editar y realizar el update para cada uno de ellos.

Tenemos una tabla de la base de datos con números de teléfono y nombres de contacto. En este caso, para simplificar, vamos a editar los registros de 10 en 10. Tendremos que mostrar un formulario con los 10 primeros teléfonos y un enlace para ver los 10 siguientes, siempre que queden registros por mostrar en la base de datos.

Podemos [ver el ejemplo en marcha](#) para hacernos una idea exacta de nuestros objetivos.

En el script vamos a tener dos partes: una en la que se muestra el formulario y otra en la que se hacen los distintos update a partir de los datos recogidos. Toda la lógica la vamos a concentrar en una única página, donde, si no se reciben datos de formulario, se muestra el formulario. Si se han recibido datos de un formulario, entonces se deben actualizar los registros.

Mostrar el formulario

Lo primero que debemos de hacer es saber a partir de qué registro se debe mostrar el formulario. Porque podemos desear mostrar los primeros 10 valores o un conjunto de registros posterior.

```
if (isset($_GET["id_mostrar"]))
    $id_mostrar = $_GET["id_mostrar"];
else
    $id_mostrar = 0;
```

Con estas líneas vemos si recibimos por GET un identificador a partir del que mostrar los registros. Si no se recibe nada, se entiende que se desea recibir los registros desde el principio (id=0).

Para mostrar el formulario tenemos que recibir de la base de datos los registros a mostrar.

Para ello se realiza una consulta de selección, limitando a 10 el número de registros a recibir.

```
$ssql="select * from manual_php where id>$id_mostrar limit 10";  
$result=mysql_query($ssql);
```

Ahora, haremos un formulario para editar los teléfonos. Para cada registro, se mostrará una línea con el teléfono y el nombre de la persona. Cada línea tendrá unos campos de formulario, para el identificador del registro y para el número de teléfono. El campo donde se guarda el identificador lo haremos con un hidden y el teléfono con un campo de texto. Para nombrar los campos de formulario utilizaremos un numero, para que no sean siempre los mismos, porque si se llaman todos los campos igual, a la hora de recibirlos podemos tener problemas. El código que queremos generar tendrá una forma como esta:

```
<input type=hidden name='id1' value='1'>  
Daniel  
<input type=text name='telefono1' value='0147852'>  
  
<input type=hidden name='id2' value='2'>  
Claudio  
<input type=text name='telefono2' value='9876654'>  
  
...  
  
<input type=hidden name='id10' value='88'>  
María  
<input type=text name='telefono10' value='000000'>
```

Vemos que el primer identificador tiene nombre "id1", el Segundo "id2" y así sucesivamente. Los números de teléfono también tienen campos con nombres distintos: "telefono1", "telefono2"... No vamos a dar opción a editar los nombres de contacto en este formulario, para simplificar las cosas.

Para generar el código del formulario utilizaremos un script PHP como el siguiente:

```
echo "<form action='edicion_a_la_vez.php' method=post>";  
echo "\n<table align=center>";  
echo "\n<tr><td><b>Nombre</b></td><td><b>Teléfono</b></td></tr>";  
  
$i = 1;  
while ($fila=mysql_fetch_array($result)){  
    echo "\n<input type=hidden name='id$i' value=" . $fila["id"] . ">";  
    echo "<tr>";  
    echo "<td>" . $fila["nombre"] . "</td>";  
    echo "<td><input type=text name='telefono$i' value=" . $fila["telefono"] . "></td>";  
    echo "</tr>";  
    $i++;  
$ultimo_mostrado = $fila["id"];  
}  
  
echo "\n<tr><td colspan=2 align=center><input type='submit' value='Editar todos'></td></tr>";  
echo "\n</table>";  
echo "\n</form>";
```

Por último, mostraremos un enlace para ver los siguientes registros, si es que se han mostrado registros en esta página. Utilizamos la variable \$ultimo_mostrado, para saber a partir de qué registro continuar.

```
if (isset($ultimo_mostrado))  
    echo "\n<br><a href='edicion_a_la_vez.php?id_mostrar=" . $ultimo_mostrado . ">Ver los 10 siguientes</a>";
```

Recibir los datos y hacer los UPDATE

Ahora vamos a mostrar la manera de recibir los datos del formulario y generar los update. Como tenemos hasta 10 registros por recibir, realizaremos un bucle FOR desde 1 hasta 10, con lo cual se podrán recibir todos los valores, concatenando el nombre del campo con el contador del for, desde 1 hasta 10.

Puede que en algunos casos no nos llegue el formulario con los 10 registros, porque haya menos para editar en el formulario, así que realizamos primero una comprobación para ver si están declaradas las variables que deberían llegar por POST. Por ejemplo, si está declarada la variable `$_POST["id1"]`, quiere decir que estamos recibiendo datos también en `$_POST["telefono1"]`.

```
for ($i=1;$i<=10;$i++){
    //para cada uno de los elementos que puede haber en el formulario
    if (isset($_POST["id" . $i])){
        //es que este registro estaba en el formulario
        $id = $_POST["id" . $i];
        $telefono = $_POST["telefono" . $i];
        $ssql = "update manual_php set telefono='$telefono' where id=$id";
        if (mysql_query($ssql))
            echo "<br>Teléfono actualizado con éxito";
        else
            echo "<br>Teléfono NO actualizado";
    }
}
```

En definitiva, se hace un bucle de 1 a 10 para recibir los datos que haya, si es que nos han llegado del formulario. Para cada registro que hemos detectado que nos llega por el formulario, se genera y ejecuta el update correspondiente, con lo que se irán actualizando los valores de la tabla.

El código completo

Mostramos el código completo de este ejemplo, para verlo de una manera global y asegurarnos que todo el mundo lo copia perfectamente:

```
<HTML>
<HEAD>
<TITLE>edicion_a_la_vez.php</TITLE>
</HEAD>
<BODY>
<h1 align="center">Edición de varios registros a la vez</h1>
<br>
<br>
<?
//Conexion con la base
require ($raiz . "../..../librerias/principales.php");
$conn = mysql_conexion();

if (!$_POST){
    //si no recibo datos de POST, muestro el formulario

    //es posible que recibamos un id a partir del que hay que mostrar los datos
    if (isset($_GET["id_mostrar"]))
        $id_mostrar = $_GET["id_mostrar"];
    else
        $id_mostrar = 0;
```

```
//extraemos de la base de datos los registros a mostrar
//Ejecutamos la sentencia SQL, limitando la búsqueda a 10 registros
$ssql="select * from manual_php where id>$id_mostrar limit 10";
$result=mysql_query($ssql);

echo "<form action='edicion_a_la_vez.php' method=post>";
echo "\n<table align=center>";
echo "\n<tr><td><b>Nombre</b></td><td><b>Teléfono</b></td></tr>";

$i = 1;
while ($fila=mysql_fetch_array($result)){
    echo "\n<input type=hidden name='id$i' value='' . $fila["id"] . "'>";
    echo "<tr>";
    echo "<td>" . $fila["nombre"] . "</td>";
    echo "<td><input type=text name='telefono$i' value='' . $fila["telefono"] . "'></td>";
    echo "</tr>";
    $i++;
    $ultimo_mostrado = $fila["id"];
}

echo "\n<tr><td colspan=2 align=center><input type='submit' value='Editar todos'></td></tr>";
echo "\n</table>";
echo "\n</form>";

//si se han mostrado registros, pongo el enlace para ver los siguientes
if (isset($ultimo_mostrado))
    echo "\n<br><a href='edicion_a_la_vez.php?id_mostrar=" . $ultimo_mostrado . "'>Ver los 10 siguientes</a>";
}else{

//es que he recibido datos de formulario, entonces tengo que recibirlos y actualizar la base de datos
for ($i=1;$i<=10;$i++){
    //para cada uno de los elementos que puede haber en el formulario
    if (isset($_POST["id" . $i])){
        //es que este registro estaba en el formulario
        $id = $_POST["id" . $i];
        $telefono = $_POST["telefono" . $i];
        $ssql = "update manual_php set telefono='$telefono' where id=$id";
        if (mysql_query($ssql))
            echo "<br>Teléfono actualizado con éxito";
        else
            echo "<br>Teléfono NO actualizado";
    }
}
echo "\n<p><a href=edicion_a_la_vez.php>Volver</a>";
}
?>
</BODY>
</HTML>
```

Para finalizar, podemos [ver el ejemplo en marcha](#) en una página aparte.

*Artículo por **Miguel Angel Alvarez***

SiteMaps de Google en PHP

Antes de poner manos a la obra vamos a hablar un poco de Google SiteMaps.

Google Sitemaps permite a los webmasters añadir en un XML las páginas que quieren que

estén en el index de Google. La participación en el programa es gratuita, la inclusión no está garantizada, pero Google piensa que de este modo podrá llegar a indexar un mayor número de páginas que con el simple rastreo tradicional.

El XML también dejará que los webmasters indiquen cada cuanto se actualiza su página o cada cuanto tiempo quiere que sea revisitado.

Como funciona el nuevo programa?

Los webmasters crean un archivo XML con las URLs que quiere sean rastreadas, así como una pequeña anotación por URL indicando la última actualización y el ritmo de actualización de la página. Google alojará el Sitemap en sus servidores indicandote exactamente donde se encuentra.

Solo es necesario utilizar una cuenta de Google y registrarte de manera gratuita.

Este script toma el directorio raíz de nuestro sitio web, lo lee y lista todos aquellos archivos cuya extensión sea .php, pero ustedes pueden cambiarlo para que liste lo que ustedes deseen, en base a sus necesidades, esto es sólo un ejemplo.

Pero basta de palabras y vamos a ver el código

```
<?php
if ($gestor = opendir('.')) { //Abrimos el directorio donde estamos parados, aca pueden darle el directorio que deseen

echo "<" . "?xml version=\"1.0\" encoding=\"UTF-8\" . "?" . ">"; //etiqueta de inicio de un archivo XML, parseado
para no tener problemas con las etiquetas

/* Datos para el sitemaps de Google, chequea esta parte en
https://www.google.com/webmasters/sitemaps/docs/es/overview.html */
?>

<urlset xmlns="http://www.google.com/schemas/sitemap/0.84">

    <url>
        <loc>http://www.tusitio.com/</loc>
        <lastmod>2006-03-23</lastmod>
        <changefreq>monthly</changefreq>
        <priority>0.8</priority>
    </url>

<?
// empezamos a generar la iteracion
while (false !== ($archivo = readdir($gestor))) {
    $p = explode(".", $archivo);

    if(strtolower($p[count($p)-1]) == "php")
    {
        echo "
<url>
        <loc>http://www.tusitio.com/$archivo</loc>
        <lastmod>2006-03-23</lastmod>
        <changefreq>yearly</changefreq>
    </url>";
    }
}

//cerramos
closedir($gestor);
echo "
```

```
</urlset>";  
}  
?>
```

Demasiado Fácil no? claro que si queremos darle datos desde la base de datos, cosas como contenido.php?id=1 se empieza a complicar un poco la cosa, pero ni tanto, de todas maneras, eso lo desarrollas tu en base, como ya dije, a tus necesidades.

Artículo por [Juan Edgardo Jorquera Uribe](#)

Creación de BBcode en PHP

Veremos cómo realizar esto paso a paso. Lo primero es crear la función PHP sin incluir sus instrucciones.

```
<?  
function BBcode($texto){  
// aquí ira el contenido de la función  
return $texto;  
}  
?>
```

Para realizar esto vamos a utilizar un función muy básica en PHP, [preg_replace\(\)](#).

De tal forma que si queremos que cuando en **\$texto** aparezcan los tags [b] y [/b] se conviertan en el HTML y podríamos hacerlo de la siguiente manera:

```
<?  
preg_replace("/\[b\](.*?)\[\/b\]/is", "<b>$1</b>", $texto);  
?>
```

Y así podríamos realizarlo con todos los tags sencillos que deseemos pero como siempre hay una mejor opción y es utilizar un *array* para las expresiones regulares(patcón) y otro para el texto de reemplazo:

```
<?  
function BBcode($texto){  
    $a = array(  
        "\[i\](.*?)\[\/i\]/is",  
        "\[b\](.*?)\[\/b\]/is",  
        "\[u\](.*?)\[\/u\]/is"  
    );  
  
    $b = array(  
        "<i>$1</i>",  
        "<b>$1</b>",  
        "<u>$1</u>"  
    );  
    $texto = preg_replace($a, $b, $texto);  
    return $texto;  
}  
?>
```

Bien ya tenemos una sencilla función para poder poner texto en cursiva, negrita y tachado.

Pero lo propuesto es también poder poner imágenes y enlaces, para las imágenes es muy similar a lo anterior, pero para los enlaces es un poco mas complicado, la función quedaría de la siguiente manera:

```
<?
function BBcode($texto){
    $a = array(
        "\[i](.*?)\[\/i]/is",
        "\[b](.*?)\[\/b]/is",
        "\[u](.*?)\[\/u]/is",
        "\[img](.*?)\[\/img]/is",
        "\[url=(.*?)](.*?)\[\/url]/is"
    );

    $b = array(
        "<i>$1</i>",
        "<b>$1</b>",
        "<u>$1</u>",
        "<img src=\"\$1\" />",
        "<a href=\"\$1\" target=\"_blank\">$2</a>"
    );
    $texto = preg_replace($a, $b, $texto);
    return $texto;
}
?>
```

Ahora bien podemos añadir a la función alguna función extra como puede ser que el texto enviado en html no sea permitido y que este se muestre, esto lo arreglamos con la [htmlentities\(\)](#) y también podemos agregar que los saltos de línea sean convertidos automáticamente en
 con [nl2br\(\)](#), nuestra función **BBcode()** final quedará de la siguiente forma:

```
<?
function BBcode($texto){
    $texto = htmlentities($texto);
    $a = array(
        "\[i](.*?)\[\/i]/is",
        "\[b](.*?)\[\/b]/is",
        "\[u](.*?)\[\/u]/is",
        "\[img](.*?)\[\/img]/is",
        "\[url=(.*?)](.*?)\[\/url]/is"
    );
    $b = array(
        "<i>$1</i>",
        "<b>$1</b>",
        "<u>$1</u>",
        "<img src=\"\$1\" />",
        "<a href=\"\$1\" target=\"_blank\">$2</a>"
    );
    $texto = preg_replace($a, $b, $texto);
    $texto = nl2br($texto);
    return $texto;
}
?>
```

Artículo por **Mario Juárez**

¿Qué podemos hacer con XML y PHP ?

Aunque utilizar estas dos tecnologías juntas no excluye usar bases de datos, iese es lo mejor de todo!.

En la segunda parte de este artículo veremos con juntar estas tres tecnologías para darle rienda suelta a nuestra imaginación.

Vamos al lío.

Para empezar vamos a crear nuestro archivo XML de ejemplo, al que llamaremos "noticias.xml" (por ser un poquito originales ;-)):

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bloque>
<noticia>
<titulo>Hola Caracola </titulo>
<autor>KaoS</autor>
<cuerpo>Olla Kaitos a Luisete</cuerpo>
</noticia>
<noticia>
<titulo>Nuevo articulo en desarrolloweb </titulo>
<autor>Raul</autor>
<cuerpo>Jeje hola, aqui estamos </cuerpo>
</noticia>
</bloque>
```

Bueno ya tenemos creado nuestro archivo XML, que como ya sabemos nos permite crear nuestras propias etiquetas, aunque según en que estemos trabajando será recomendable seguir los estándares establecidos por el [w3c](#).

Ahora necesitamos crear un archivo PHP que lea nuestro archivo "noticias.xml". ¿Como podemos hacer esto? , pues es muy simple, porque PHP ya incluye ciertas funciones para el trabajo con archivos XML. Nosotros vamos a utilizar unas funciones que trabajan sobre PHP 4 ya que aún no todo el mundo tiene PHP 5 en su servidor, pero la forma de trabajar es muy similar. Dejo aquí una referencia de como trabajar del mismo modo pero usando las funciones de [PHP 5](#).

Vamos a trabajar.

Lo primero que tenemos que hacer es leer el archivo, para ello utilizaremos nuestra queridísima función [fopen](#). Da igual si el archivo se encuentra en nuestro servidor o no, por lo que si nos interesa podríamos crear un archivo PHP que funcionara igualmente en un servidor remoto que leyese las noticias de nuestra web.

```
//$ruta_fichero="http://www.dominio.com/noticias.xml";
$ruta_fichero="noticias.xml";

$contenido = "";
if($da = fopen($ruta_fichero,"r"))
{
while ($aux= fgets($da,1024))
{
$contenido.=$aux;
}
fclose($da);
}
```

```
else
{
echo "Error: no se ha podido leer el archivo <strong>$ruta_fichero</strong>";
}
```

Si todo ha ido correctamente ahora tendremos nuestro fichero XML cargado en nuestra variable `$contenido`. Ahora un detalle, debido a que nuestras noticias podrían tener caracteres especiales, para evitar fallos le meteremos un pequeño filtro, que en este caso por ejemplo vamos a sustituir las tildes y las eñes en el caso de que las hubiesen, para ello utilizaremos la función [ereg_replace](#).

```
$contenido=ereg_replace("á","a",$contenido);
$contenido=ereg_replace("é","e",$contenido);
$contenido=ereg_replace("í","i",$contenido);
$contenido=ereg_replace("ó","o",$contenido);
$contenido=ereg_replace("ú","u",$contenido);
$contenido=ereg_replace("Á","A",$contenido);
$contenido=ereg_replace("É","E",$contenido);
$contenido=ereg_replace("Í","I",$contenido);
$contenido=ereg_replace("Ó","O",$contenido);
$contenido=ereg_replace("Ú","U",$contenido);
$contenido=ereg_replace("Ñ","NI",$contenido);
$contenido=ereg_replace("ñ","ni",$contenido);
```

El siguiente paso es cargar nuestro archivo XML en una estructura que podamos trabajar con PHP de forma cómoda, para esta tarea vamos a utilizar las funciones dom que vienen implementadas a partir de la versión 4 de PHP. Concretamente usaremos:

- [domxml_open_mem](#) : Crea un objeto DOM desde un documento XML
- [document_element](#) : Crear un nuevo nodo de tipo elemento
- [get_elements_by_tagname](#): Obtiene elementos por el nombre de etiqueta
- [get_content](#) : Obtiene el contenido del nodo

```
$tagnames = array ("titulo","autor","cuerpo");

if (!$xml = domxml_open_mem($contenido))
{
echo "Ha ocurrido un error al procesar el documento<strong> \"$ruta_fichero\"</strong> a XML <br>";
exit;
}
else
{
$raiz = $xml->document_element();

$tam=sizeof($tagnames);

for($i=0; $i<$tam; $i++)
{
$nodo = $raiz->get_elements_by_tagname($tagnames[$i]);
$j=0;
foreach ($nodo as $etiqueta)
{
$matriz[$j][$tagnames[$i]]=$etiqueta->get_content();
$j++;
}
}
}
```

Analizamos más detenidamente este último trozo de código a ver que es lo que realmente hace. Para empezar hemos creado un array con los campos que contiene cada noticia en la variable "tagnames". A continuación cargamos la variable contenido en un objeto DOM, en el

caso de que todo haya ido bien extraemos el nodo raiz, en nuestro caso "bloque". El siguiente paso es calcular el numero de campos que obtendremos de cada noticia, para ello utilizamos la función [sizeof](#) que nos devuelve el tamaño del array.

Es ahora cuando extraemos la verdadera información del documento XML. Esta información la vamos a introducir en una matriz para que nos sea más simple trabajar con los datos. De forma que matriz quedase algo así:

indice \ Nombre Columna	titulo	autor	cuerpo
0	Hola Caracola	KaoS	Olla Kaitos a Luisete
1	Nuevo articulo en desarrolloweb	Raul	Jeje hola, aqui estamos

El primer bucle extrae las etiquetas de los nodos (primero titulo, despues autor y luego cuerpo).

El foreach se encarga de sacar una a una las etiquetas de cada una de las noticias, por lo que primero extrae "Hola Caracola" y en la segunda iteración "Nuevo articulo en desarrolloweb". De este modo vamos guardando en nuestra matriz los datos extraidos.

En la segunda iteración(repetición) del bucle for cogemos la etiqueta autor, y en el foreach extraeremos los valores para introducirlos en nuestra matriz. Y así hasta terminar. ¡Lo mejor de todo es que de esto se encarga nuestro propio bucle!, nosotros solo tendremos que preocuparnos de declarar el array de etiquetas.

Bueno para que nos sea más comodo podemos crear una función a la que le pasaremos el archivo XML que queremos que nos lea y nos devuelva una matriz con los datos, haciendo así nuestro trabajo más limpio y eficiente. El código resultante sería:

```
function CargarXML($ruta_fichero)
{
    $contenido = "";
    if($da = fopen($ruta_fichero,"r"))
    {
        while ($aux= fgets($da,1024))
        {
            $contenido.=$aux;
        }
        fclose($da);
    }
    else
    {
        echo "Error: no se ha podido leer el archivo <strong>$ruta_fichero</strong>";
    }

    $contenido=ereg_replace("á","a",$contenido);
    $contenido=ereg_replace("é","e",$contenido);
    $contenido=ereg_replace("í","i",$contenido);
    $contenido=ereg_replace("ó","o",$contenido);
    $contenido=ereg_replace("ú","u",$contenido);
    $contenido=ereg_replace("Á","A",$contenido);
    $contenido=ereg_replace("É","E",$contenido);
    $contenido=ereg_replace("Í","I",$contenido);
    $contenido=ereg_replace("Ó","O",$contenido);
    $contenido=ereg_replace("Ú","U",$contenido);
    $contenido=ereg_replace("Ñ","NI",$contenido);
    $contenido=ereg_replace("ñ","ni",$contenido);
}
```

```
$tagnames = array ("titulo","autor","cuerpo");

if (!$xml = domxml_open_mem($contenido))
{
echo "Ha ocurrido un error al procesar el documento<strong> \"$ruta_fichero\"</strong> a XML <br>";
exit;
}
else
{
$raiz = $xml->document_element();

$tam=sizeof($tagnames);

for($i=0; $i<$tam; $i++)
{
$nodo = $raiz->get_elements_by_tagname($tagnames[$i]);
$j=0;
foreach ($nodo as $etiqueta)
{
$matriz[$j][$tagnames[$i]]=$etiqueta->get_content();
$j++;
}
}

return $matriz;
}
}
```

Bueno, pues esto ya está casi todo listo, ya hemos cargado una matriz con el contenido de un archivo XML, por lo que ahora solo nos queda mostrar la información que queramos. Vamos a ver en un pequeño código como hacerlo.

```
$matriz=CargarXML("noticias.xml");

$num_noticias=sizeof($matriz);
for($i=0;$i<$num_noticias;$i++)
{
echo '
<table border=1>
<tr><td align=center>'. $matriz[$i]["titulo"].'</td></tr>
<tr><td>'. $matriz[$i]["cuerpo"].'</td></tr>
<tr><td align=right >'. $matriz[$i]["autor"].'</td></tr>
</table><br>
';
}
}
```

Voilà, ya tenemos nuestro primer ejemplo de XML+PHP. Aquí podeis ver el ejemplo funcionando: [archivo xml](#) y archivo xml formateado. Bueno esto ha sido todo por hoy, espero que os haya sido de utilidad el artículo.

*Artículo por **Raúl Jiménez Ortega***

Ordenar arrays con PHP

En este taller de PHP vamos a tratar la ordenación de arrays o tablas en PHP. Existen en PHP diversas funciones para ordenar arrays, por lo que realmente lo que vamos a ver es cómo utilizar las funciones que vienen con el lenguaje.

En el [manual de programación en PHP](#) tenemos unos artículos que explica el [array en PHP](#). Es de interesante lectura para conocer los conceptos más básicos sobre la creación y manejo de arrays en PHP.

Ahora veamos directamente las funciones disponibles para ordenar arrays.

sort()

Es la función más básica para ordenar arrays en PHP. Ordena el array de valores menores a mayores. Lo vemos con un ejemplo.

```
//Ordenar desde el menor al mayor
$alumnos = array("Pepe", "Juan", "Marcelo", "Alberto", "Gerardo");
sort($alumnos);
foreach ($alumnos as $key => $val) {
    echo "alumnos[" . $key . "] = " . $val . "<br>";
}
```

Que dará como resultado:

```
alumnos[0] = Alberto
alumnos[1] = Gerardo
alumnos[2] = Juan
alumnos[3] = Marcelo
alumnos[4] = Pepe
```

rsort()

Esta función ordena el array por valores. La "r" delante quiere decir que ordena en orden reverso, de mayor a menor.

```
//ordenar de mayor a menor (orden inverso... Reverse order)
$alumnos = array("Pepe", "Juan", "Marcelo", "Alberto", "Gerardo");
rsort($alumnos);
foreach ($alumnos as $key => $val) {
    echo "alumnos[" . $key . "] = " . $val . "<br>";
}
```

Que daría como respuesta:

```
alumnos[0] = Pepe
alumnos[1] = Marcelo
alumnos[2] = Juan
alumnos[3] = Gerardo
alumnos[4] = Alberto
```

ksort()

También podemos ordenar un array por el índice o llave, que quiere decir que en lugar de ordenar atendiendo a los valores, se ordenaría atendiendo al índice que tienen. Para ver este ejemplo utilizaremos arrays asociativos, que son los que tienen índices de texto en vez de números.

En el array siguiente vemos que tenemos índices como "h", "e", "a", en lugar de números. Lo que hará este tipo de ordenación es fijarse en esos índices para poner el array ordenado por ellos.

```
//Ordenar arrays por su índice
$calles = array("h"=>"Leganitos", "e"=>"Castellana", "a"=>"Bailén", "z"=>"Fuencarral");
ksort($calles);
foreach ($calles as $key => $val) {
    echo $key . " = " . $val . "<br>";
}
```

Esto dará como resultado esta ordenación:

a = Bailén
e = Castellana
h = Leganitos
z = Fuencarral

krsort()

También podemos ordenar por índices pero en sentido inverso. Es decir, por índices pero de mayor a menor.

```
//ordenar por índice o clave, pero en orden inverso
$calles = array("h"=>"Leganitos", "e"=>"Castellana", "a"=>"Bailén", "z"=>"Fuencarral");
krsort($calles);
foreach ($calles as $key => $val) {
    echo $key . " = " . $val . "<br>";
}
```

En este caso el resultado sería el siguiente:

z = Fuencarral
h = Leganitos
e = Castellana
a = Bailén

asort()

Esta función ordena los elementos de un array, pero manteniendo la correlación entre índices y valores a los que están asociados. Ordena por valores. Se utiliza generalmente en arrays asociativos.

```
//ordenar manteniendo los índices
$capitales = array("España" => "Madrid", "Argentina" => "Buenos Aires", "México" => "Ciudad de México", "Brasil"
=> "Brasilia");
asort($capitales);
foreach ($capitales as $key => $val) {
    echo $key . " = " . $val . "<br>";
}
```

Dará como resultado el siguiente orden de array:

Brasil = Brasilia
Argentina = Buenos Aires
México = Ciudad de México
España = Madrid

rsort()

Es lo mismo que asort(), pero realiza el orden en inverso de los valores de los arrays. Como decíamos, lo habitual es realizar este tipo de orden en arrays asociativos, pero en este caso vamos a ver el orden en un array normal (con índices numéricos) para que se vea mejor cómo trabaja la función:

```
//ordenar manteniendo los índices, Reverso
$ciudades = array("Madrid", "Barcelona", "Valencia", "Sevilla", "Bilbao");
rsort($ciudades);
foreach ($ciudades as $key => $val) {
    echo $key . " = " . $val . "<br>";
}
```

El resultado obtenido es este:

2 = Valencia
3 = Sevilla
0 = Madrid
4 = Bilbao

1 = Barcelona

natsort()

Para acabar vamos a ver esta función que hace una ordenación natural de los elementos del array, es decir, ordena tal como lo haría una persona. Hay una pequeña diferencia sobre el orden que haría sort(). La función natsort mantiene la asociación clave - valor.

```
$productos = array ("producto 11", "producto 1", "producto 12", "producto 2");
natsort($productos);
foreach ($productos as $key => $val) {
    echo $key ." = " . $val . "<br>";
}
```

Esto daría como respuesta:

```
2 = producto 1
4 = producto 2
1 = producto 11
3 = producto 12
0 = producto 20
```

Ahora, para que se vea la diferencia con un orden normal, vamos a ordenar ese mismo array con la función sort(), que hace un orden alfanumérico normal.

```
$productos = array ("producto 11", "producto 1", "producto 12", "producto 2");
sort($productos);
foreach ($productos as $key => $val) {
    echo $key ." = " . $val . "<br>";
}
```

El resultado de este orden será el siguiente:

```
0 = producto 1
1 = producto 11
2 = producto 12
3 = producto 2
```

Como se puede ver, el orden es distinto, porque en ese caso ordena por un orden alfanumérico estricto, sin tener en cuenta los valores como un humano lo haría.

Artículo por [Miguel Angel Alvarez](#)

Incluir feeds en tu web en 5 pasos

Hace poco yo quise hacer lo propio en mi sitio web de apuntes Infoapuntes.com, es decir, incluir un pequeño bloque donde mostrar titulares de noticias de informática. Sin embargo toda la información que encontré en la web circula en torno a la lectura de feeds, no a la integración en sitios web y, lo que refería a esto último hablaba de complicados códigos. Parecía que no iba a ser posible hasta que me topé con MagpieRSS.

En este manual vamos a explicar cómo incluir noticias en tu web, del medio que quieras; la fuente que hemos usado para nuestros feeds en este caso es la de Barrapunto.com y nos hemos apoyado en el script [MagpieRSS](#) para el funcionamiento.

¿Qué necesitamos?

- Conocimientos básicos de PHP.

- El script [MagpieRSS](#) para integrar RSS en tu web.
- El feed de la fuente que desees, en este caso el de Barrapunto. Este es: [Feed de Barrapunto](#)

Incluye feeds en 5 pasos

Vamos a empezar instalando el script en nuestro servidor y en nuestra web.

1. Crea una carpeta en el directorio raíz de tu servidor llamada "magpierss".
2. En el script donde vayas a publicar las noticias, incluye esto:
* `require_once "magpierss/rss_fetch.inc";`
3. A continuación, hacemos que el script procese el feed:
* `$url = http://backends.barrapunto.com/barrapunto.rss`
* `$rss = fetch_rss($url);`
4. Con esto último se creará por defecto una carpeta llamada "cache" para acelerar las llamadas al feed.
Se devuelve un objeto con la información que queremos que se encuentra en la variable `$rss->items`
5. Ahora creamos un array con la información que hemos extraído:
* `$items = array_slice($rss->items, 0);`

Ya tenemos los siguientes datos, que pueden ser accedidos como cualquier variable:

- `$items[$num_articulo]['title']` -> Título de la noticia/artículo.
- `$items[$num_articulo]['summary']` -> Resumen de la noticia.
- `$items[$num_articulo]['pubdate']` -> Fecha de publicación.
- `$items[$num_articulo]['link']` -> Enlace al feed en la página original.
- `$items[$num_articulo]['author']` -> Autor del contenido.

donde `$num_articulo` es el número de artículo al que queremos acceder. Están ordenados por la fecha de publicación, donde el más reciente es el número 0 (cero).

Por último, lo que queda es que apliques tu creatividad para darle formato a los resultados, incluyéndolos en una marquesina o lo que se te ocurra. Incluso podrías usar [Cron](#) para ejecutar los scripts periódicamente.

Ejemplo

Aquí tenemos el ejemplo de un script completo:

```
require_once "magpierss/rss_fetch.inc";

//barrapunto
$url = "http://backends.barrapunto.com/barrapunto.rss";

$rss = fetch_rss($url);

$items = array_slice($rss->items, 0);

$max_noticias = 30;
$cont = 0;
echo '<h1> Titulares</h1>';
echo '<marquee scrollamount="1" direction="up" loop="true" onmouseover="this.stop()" onmouseout="this.start()" align="left">'; while(!empty($items[$cont])&&($cont<$max_noticias)){

echo '<b>Autor: </b> '.$items[$cont]["author"].'<br>';
echo '<b>Fecha: </b> '.$items[$cont]["pubdate"].'<br>';
```

```
echo '<a href="'. $items[$cont]["link"].'" target="_blank">'. $items[$cont]["title"].'</a><br>';  
echo $items[0]["pubdate"].'<br>';  
echo $items[0]["summary"].'<br>';  
$cont++;  
}  
echo '</marquee>';
```

Conclusión

Hasta aquí este manual sobre la inclusión de feeds en tu web. Comprobaréis que a partir de ahora, la inclusión de contenidos en tu web es tremendamente fácil y que además, tienes un amplísimo abanico de contenidos de todo tipo de categorías. Algunos feeds interesantes en castellano:

Medio	Descripción	Feed
barrapunto.com	Noticias y artículos de Informática	http://backends.barrapunto.com/barrapunto.rss
elpais.es	Prensa diaria	http://www.elpais.es/static/rss/index.html (web con todos los feeds)
elmundo.es	Prensa diaria	http://rss.elmundo.es/rss (web con todos los feeds)
alzado.org	Desarrollo web	http://www.alzado.org/xml/alzado.xml
marca.es	Prensa deportiva	http://www.marca.com/rss/

[Página oficial de MagpieRSS](#)

Licencia

Es muy importante que, antes de incluir cualquier contenido en tu web, revises la licencia y las condiciones bajo las que se distribuyen los contenidos. En el caso de Barrapunto, la licencia es esta: "[Licencia de Creative Commons](#)"

*Artículo por **Raúl Cano***

Calcular días entre dos fechas con PHP

A veces necesitamos saber los días que han transcurrido entre dos fechas. Con PHP podemos hacer esa tarea fácilmente, simplemente restando el valor timestamp de las dos fechas y convirtiendo a días. Lo explicaremos paso a paso en este taller.

El ejercicio es muy sencillo. Vamos a obtener los valores timestamp de las dos fechas. (Recordar que los timestamp son los segundos que han pasado desde las cero horas del 1 de enero de 1970) Como los dos timestamps son una cantidad de segundos, no tenemos más que restarlos para obtener los segundos de diferencia entre las dos fechas. Luego se trataría de convertir esos segundos en días para obtener el dato que estamos buscando.

Veamos entonces la manera de obtener un timestamp de una fecha. Entre las funciones de fechas de PHP hay varias que nos pueden servir para trabajar con timestamp, pero nosotros tenemos que utilizar una en concreto llamada mktime(). Esta función recibe varios parámetros:

```
mktime ( [int hora [, int minuto [, int segundo [, int mes [, int dia [, int año [, int es_dst]]]]]] )
```

El primer parámetro es la hora, luego los minutos y segundos. Luego los meses, días y años. Con todos esos valores nos devuelve el timestamp de una fecha cualquiera. Podemos omitir parámetros y en ese caso tomará los valores de la fecha actual del servidor.

El código para obtener los timestamp de un par de fechas inventadas podría ser algo como el siguiente:

```
//defino fecha 1
$ano1 = 2006;
$mes1 = 10;
$dia1 = 2;

//defino fecha 2
$ano2 = 2006;
$mes2 = 10;
$dia2 = 27;

//calculo timestam de las dos fechas
$timestamp1 = mktime(0,0,0,$mes1,$dia1,$ano1);
$timestamp2 = mktime(0,0,0,$mes2,$dia2,$ano2);
```

Luego, podríamos restar los timestamp y convertir los segundos en días:

```
//resto a una fecha la otra
$segundos_diferencia = $timestamp1 - $timestamp2;
//echo $segundos_diferencia;

//convierto segundos en días
$dias_diferencia = $segundos_diferencia / (60 * 60 * 24);
```

Para convertir los segundos en días, como se ha podido observar en el código, hay que dividir entre el número de segundos de un día. (60 segundos de un minuto, por los 60 minutos de una hora, por las 24 horas de un día).

Ahora bien, con un código como el anterior, el valor de los días de diferencia puede tener decimales y ser negativo. Nosotros queremos un número de días entero y positivo. Entonces todavía tendremos que hacer un par de operaciones matemáticas. Primero quitar el signo negativo y luego quitar los decimales.

```
//obtengo el valor absoluto de los días (quito el posible signo negativo)
$dias_diferencia = abs($dias_diferencia);

//quito los decimales a los días de diferencia
$dias_diferencia = floor($dias_diferencia);
```

Los decimales los quitamos simplemente redondeando hacia abajo. Puesto que si tenemos un número decimal de días no ha llegado a un día completo y no nos interesa contabilizarlo.

El código completo se puede ver a continuación:

```
<?
//defino fecha 1
$ano1 = 2006;
$mes1 = 10;
```

```
$dia1 = 2;

//defino fecha 2
$ano2 = 2006;
$mes2 = 10;
$dia2 = 27;

//calculo timestam de las dos fechas
$timestamp1 = mktime(0,0,0,$mes1,$dia1,$ano1);
$timestamp2 = mktime(4,12,0,$mes2,$dia2,$ano2);

//resto a una fecha la otra
$segundos_diferencia = $timestamp1 - $timestamp2;
//echo $segundos_diferencia;

//convierto segundos en días
$dias_diferencia = $segundos_diferencia / (60 * 60 * 24);

//obtengo el valor absoluto de los días (quito el posible signo negativo)
$dias_diferencia = abs($dias_diferencia);

//quito los decimales a los días de diferencia
$dias_diferencia = floor($dias_diferencia);

echo $dias_diferencia;
?>
```

Artículo por [Miguel Angel Alvarez](#)

Ejemplo de conexión con base de datos Access en PHP

PHP dispone de diversos juegos de funciones para conectar con distintos tipos de bases de datos. Por ejemplo, existe un juego de funciones específico para MySQL, otro para Oracle, PostgreSQL, etc. Aunque no existe entre sus librerías ninguna función para acceder específicamente a una base de datos Access. Para ello, disponemos de las funciones de conexión ODBC con bases de datos.

ODBC es un estándar de conexión con bases de datos que utilizan los sistemas Windows. Con ODBC se puede acceder a cualquier base de datos, siempre que exista el correspondiente driver ODBC para esa base de datos. Las conexiones ODBC son propias de sistemas Microsoft, por lo que podremos utilizarlas desde cualquier lenguaje de programación en Windows, como PHP, para acceder a cualquier base de datos, incluida Access.

Lo primero que tendremos que hacer para conectar con PHP es crear un DSN en nuestro ordenador. Un DSN es un nombre de conexión que utilizaremos para referenciar a una base de datos. Podemos tener tantos DSN en nuestro sistema como necesitemos, para acceder desde aplicaciones o lenguajes de programación a bases de datos por ODBC. Esto lo hacemos desde el panel de control de Windows en la opción "Herramientas administrativas - Orígenes de datos ODBC". Tenemos que crear un DNS de sistema o de usuario, en la solapa correspondiente y luego apretando agregar. Habrá que seleccionar el driver ODBC que tenemos que utilizar, como se trata de una base de datos de Access, debemos seleccionar el driver ODBC de Access. Luego habrá que darle un nombre al DSN y seleccionar el archivo .mdb que contiene nuestra base de datos. Una vez realizado esto, necesitaremos acordarnos del nombre del DSN que hemos creado, porque lo tendremos que utilizar en nuestro código de conexión.

Referencia: Se explica como crear un DSN con más detalle en el artículo [Pasos previos II: Conexión a](#)

BD. Es un capítulo del manual de ASP, pero se hace igual para conectar Access con PHP.

Si tenemos nuestra página alojada en un proveedor de hosting y queremos conectar Access con PHP, si es que lo permiten (pues lo típico en esos casos es utilizar una base de datos MySQL), tendríamos que preguntarles cómo crear el DSN en sus sistemas.

Una vez tengamos el DSN utilizaremos un código de conexión como este:

```
$conn_access = odbc_connect ("guiarte_access", "", "");
```

Con la función `odbc_connect()` se conecta con una base de datos por ODBC, indicando el nombre del DSN en el primer parámetro y luego el usuario y la contraseña. En Access normal es que no se tenga siquiera usuario y contraseña, por lo que esos dos parámetros los he pasado con una cadena vacía. Nos devuelve el identificador de la conexión con la base de datos.

Para ejecutar una sentencia SQL utilizaremos la función `odbc_exec()`. De este modo:

```
$rs_access = odbc_exec ($conn_access, "select * from tabla");
```

Como se puede ver, `odbc_exec()` recibe el identificador de la conexión obtenido anteriormente y la sentencia SQL a ejecutar. Devuelve un identificador del conjunto de registros resultado de ejecutar la sentencia.

Luego utilizaremos funciones como `odbc_fetch_array()` o `odbc_fetch_object()` para acceder a cada uno de los registros obtenidos en el conjunto de resultados.

Un código completo de conexión con una base de datos Access podría ser el siguiente:

```
if ($conn_access = odbc_connect ("guiarte_access", "", "")){  
    echo "Conectado correctamente";  
    $ssql = "select * from libros";  
    if($rs_access = odbc_exec ($conn_access, $ssql)){  
        echo "La sentencia se ejecutó correctamente";  
        while ($fila = odbc_fetch_object($rs_access)){  
            echo "<br>" . $fila->titulo;  
        }  
    }else{  
        echo "Error al ejecutar la sentencia SQL";  
    }  
} else{  
    echo "Error en la conexión con la base de datos";  
}
```

Nota: Las funciones `odbc_fetch_array()` y `odbc_fetch_object()`, así como otras funciones de PHP para el acceso por ODBC a bases de datos, son idénticas o muy similares en su utilización a las funciones propias para otras bases de datos como MySQL. En nuestros [manuales y talleres de PHP](#) tenemos muchos ejemplos de conexión y trabajo con bases de datos MySQL, que nos pueden servir de guía para aprender a trabajar con bases de datos Access. Simplemente habría que tener el DSN para la conexión con la base de datos Access y cambiar los nombres de las funciones como `mysql_fetch_array()` por sus correspondientes, como `odbc_fetch_array()`.

Artículo por [Miguel Angel Alvarez](#)

Utilizar Curl para copiar una imagen de una web en nuestro disco duro

Vamos a realizar un ejemplo complejo de utilización de PHP, en el que nos conectamos con un servidor web para recibir un archivo de imagen que está alojado en dicho servidor. Para especificar la imagen que queremos extraer utilizamos la URL y luego especificamos otra ruta

dentro de nuestro disco duro, donde copiaremos la imagen que hemos recibido de dicha URL.

Yo he utilizado este código para facilitar la migración automática de un sitio web. El sitio web nuevo tenía una estructura distinta y por eso se tenían que recibir las imágenes para guardarlas en otros directorios. Con esta función, y otras similares, he podido extraer todas las imágenes del sitio web antiguo y colocarlas en los directorios correctos en el sitio web nuevo.

Como decía, vamos a utilizar CURL para realizar esta tarea. Seguro que existen otras maneras de llevarla a cabo, pero esta me ha parecido bastante práctica. Curl es una librería para trabajo y tratamiento de información en URLs, es decir, para hacer cosas con direcciones URL, como explorar el contenido que tienen, copiar el contenido a otros lugares, comprobar la existencia de una URL. He de admitir que no controlo mucho la librería Curl, pero podría decir que es bastante potente y compleja.

CURL se encuentra en PHP

Según la instalación que tengamos de PHP tendremos disponibles, o no, las funciones de la librería CURL. La mejor manera de saber si disponemos de Curl en nuestra instalación PHP es invocar una de sus funciones. Si nos muestra un error de función no existente, es que no tenemos posibilidad de usar Curl. El propio manual de PHP especifica qué debemos hacer para poder utilizar las librerías.

Para poder usar estas funciones, se debe compilar PHP añadiendo el parámetro `--with-curl[=DIR]`, donde DIR apunta al directorio que contiene los directorios lib y include de la librería. En el directorio include, debe existir una carpeta llamada "curl" y que contiene los archivos easy.h y curl.h. Además, debe existir un archivo llamado libcurl.a en el directorio "lib". A partir de la versión de PHP 4.3.0 se puede configurar que PHP haga uso de CURL para el manejo de las conexiones con URLs, mediante la opción `--with-curlwrappers`.

Nota para los usuarios de plataformas Windows: Para activar este módulo en entornos Windows, se deben copiar los archivos libeay32.dll y ssleay32.dll que se encuentran en la carpeta DLL del directorio PHP/Win32 a la carpeta SYSTEM de Windows, que normalmente se encuentra en C:\WINNT\SYSTEM32 o C:\WINDOWS\SYSTEM.

No obstante, en mi ordenador local utilizo [Easy PHP](#), que es un paquete que te instala el conjunto Apache-PHP-MySQL en tu sistema Windows, sin que tengas que hacer nada, ni aprender a configurar los distintos sistemas. Easy PHP no dispone por defecto de soporte para Curl, pero cambiando una línea del archivo de configuración de PHP (el archivo php.ini) podremos dar soporte a Curl. La línea en concreto es:

```
;extension=php_curl.dll
```

Esta línea en principio está comentada, por eso empieza por ";". Simplemente habrá que quitar el "punto y coma" para que EasyPHP disponga de soporte Curl. Aunque no utilices EasyPHP prueba a descomentar esta línea, si es que no dispones de soporte para CURL.

Función para obtener una imagen de una URL determinada

Vamos a tratar ya el objetivo final del artículo, que es traerse una imagen alojada en una web a nuestro disco duro. Para ello hemos creado una función que recibe dos parámetros. El primero es la URL de la imagen que deseamos obtener y el segundo es el nombre del archivo que queremos generar en el disco duro, donde copiaremos la imagen. El nombre del archivo destino es relativo al documento, pudiendo indicar una ruta relativa, compuesta por varios directorios y un nombre de archivo.

```
function recibe_imagen ($url_origen,$archivo_destino){
$mi_curl = curl_init ($url_origen);
$fs_archivo = fopen ($archivo_destino, "w");
curl_setopt ($mi_curl, CURLOPT_FILE, $fs_archivo);
curl_setopt ($mi_curl, CURLOPT_HEADER, 0);
curl_exec ($mi_curl);
curl_close ($mi_curl);
fclose ($fs_archivo);
}
```

En esta función se da de alta una sesión CURL, con `curl_init($url_origen)`, en donde pasamos la URL a la que deseamos acceder.

Luego utilizamos las funciones del sistema de archivos de PHP para crear un nuevo archivo en el disco duro del ordenador. Si nos fijamos, se indica el archivo destino y el parámetro "w" que quiere decir que la conexión con el archivo es para escritura.

Luego indicamos un par de opciones en la sesión CURL.

```
curl_setopt ($mi_curl, CURLOPT_FILE, $fs_archivo);
curl_setopt ($mi_curl, CURLOPT_HEADER, 0);
```

La más importante es la primera, en la que se indica el valor `CURLOPT_FILE`, donde se asigna el archivo en el que se va a guardar los datos de la URL. El parámetro `$fs_archivo` debe ser un recurso de tipo stream o flujo de datos. En este caso es el archivo destino, conectado con el sistema de archivos del servidor anteriormente en la misma función.

La segunda opción definida hace que el encabezamiento no se incluya en la salida.

Para continuar, ejecutamos la conexión CURL con `curl_exec()` y por último, se cierran tanto la conexión CURL como el archivo donde hemos guardado la imagen.

Conclusión

CURL es una librería muy potente. Esta es sólo una de las utilidades de la librería, que puede servir de ejemplo para entender el uso y dar pie a otras utilidades interesantes. Podría haberse complicado un poco el script simplemente comprobando que la URL indicada en el parámetro es correcta o realizando otro tipo de acciones de validación o tratamiento de la imagen recibida en la URL.

Artículo por [Miguel Angel Alvarez](#)

Control de la salida en PHP

Como sabemos, PHP realiza un procesamiento de la página y envía al ordenador del usuario el resultado de procesar el código PHP. Por regla general, a medida que va procesando la página, se envía el código HTML resultante al cliente, pero esta configuración se puede cambiar, incluso en tiempo de ejecución.

Con PHP podemos almacenar la salida, a medida que se va generando, en un buffer. De modo que no se envíe ningún dato al cliente hasta que se indique expresamente. Existen una serie

de funciones que se utilizan para conseguir este comportamiento, que son las [funciones de control de salida](#).

Este comportamiento es muy útil cuando se tiene que enviar información en la cabecera de la página, después de haber empezado a procesar el código PHP y haber empezado a generar la salida.

Nota: Como debemos saber, existen funciones como `header()` o `setcookie()` que deben ejecutarse antes de haber enviado ningún texto de la página al cliente. En caso contrario se producirá un error "Cannot modify header information - headers already sent by..."

En este artículo veremos un pequeño ejemplo de código PHP que realiza el buffering de la salida, para enviarla una vez se ha terminado de procesar todo el código.

Utilizaremos dos funciones que posiblemente no conozcamos, para el control de la salida: `ob_start()` y `ob_end_flush()`.

La función `ob_start()` sirve para indicarle a PHP que se ha de iniciar el buffering de la salida, es decir, que debe empezar a guardar la salida en un bufer interno, en vez de enviarla al cliente. De modo que, aunque se escriba código HTML con `echo` o directamente fuera del código PHP, no se enviará al navegador hasta que se ordene explícitamente. O eventualmente, hasta que se acabe el procesamiento de todo el archivo PHP.

La función `ob_end_flush()` sirve para indicar a PHP que se desea realizar el volcado de todo el bufer en la salida, con lo que se enviará al cliente que ha solicitado la página.

Veamos este código:

```
<?
ob_start();
echo "<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">"
?>
<html>
<head>
<title>Página procesada con buffer de salida</title>
</head>
<body>
Esta es mi página!!!
</body>
</html>
<?
setcookie("nombre", "pepe");
ob_end_flush();
?>
```

Nada más comenzar se ejecuta `ob_start()`. Con esto se guardará toda la salida en un buffer. Ahora, cuando se escribe en la página, en la siguiente línea con el `echo`, y en las otras, fuera del código PHP, lo único que ocurre es que se va llenando el mencionado buffer.

Antes de terminar la página, en el siguiente bloque de código PHP, se envía una cookie al navegador del usuario. Esa cookie llega sin problemas y no genera ningún error, a pesar que se ha escrito código de la página, dado que el código no se había enviado al navegador, sino que se había almacenado en el buffer.

Por último, se ejecuta `ob_end_flush()` para enviar el buffer a la salida.

Podemos probar a comentar las líneas que ejecutan las funciones `ob_start()` y `ob_end_flush()`. Entonces veríamos como la función `setcookie()` provocaría un error, porque esta función no se puede ejecutar si ya se ha escrito texto en la página y por tanto se han enviado ya las cabeceras del http al cliente.

Es un ejemplo muy sencillo del control de la salida en PHP. Pero a partir de aquí se puede complicar todo lo que se necesite. Esperamos que este artículo sea una interesante introducción para continuar investigando temas relacionados con esta funcionalidad básica de PHP.

Puede resultarte interesante seguir con este tema, por lo que te recomiendo que leas el artículo [Control de salida en PHP II](#).

Artículo por Miguel Angel Alvarez

Control de salida en PHP II

Continuamos explorando algunas utilidades interesantes de PHP para realizar control de salida, a través de un buffer, para no enviar al cliente el código HTML generado por PHP hasta que el programador lo desee.

Este artículo continúa con otro anterior publicado en DesarrolloWeb.com: [Control de salida en PHP](#).

Ahora vamos a aprender a eliminar o cancelar una salida almacenada en un buffer y a ejecutar funciones para realizar un procesamiento del código almacenado en un buffer antes de enviarlo al cliente. Para ello vamos a conocer nuevas funciones:

La función `ob_clean()` limpia el buffer de salida. Eso quiere decir que lo que tengamos en el bufer de salida se perderá.

La función `ob_end_clean()` sirve para limpiar el buffer de salida y además deshabilitarlo. Es decir, termina de trabajar con el bufer y además descarta todos los cambios que se hubieran incluido en el bufer.

También utilizaremos un parámetro que no habíamos visto de la función `ob_start(parametro)`. Ese parámetro sirve para indicar el nombre de una función que será llamada antes de enviar el buffer de salida al cliente.

Veamos este código, que hace uso de las nuevas funciones y lo vamos explicando luego paso a paso.

```
<html>
<head>
  <title>Página procesada con buffer de salida</title>
</head>
<body>
<?
//inicio el buffer de salida
ob_start();
```

```
echo "hola!!! esto se va al buffer!";

//limpio el buffer de salida
ob_clean();

echo "Otra vez escribo!!";

//limpio el buffer de salida y lo deshabilito
ob_end_clean();

echo "Esto si que va a aparecer en la página<br>";

function convierte_caracteres_especiales($buffer){
    return htmlentities ($buffer);
}

//inicio el buffer de salida
ob_start("convierte_caracteres_especiales");

echo "Tenía que probar más cosas. Mañana espero que se lea con interés.";

ob_end_flush();
?>
</body>
</html>
```

Con `ob_start()`; iniciamos el buffering de la salida. A partir de ahora todo lo que se escriba en la página se guardará en el buffer. Por lo tanto, el siguiente `echo "hola!!! esto se va al buffer!";` se guardará en el buffer.

Con la línea `ob_clean()`; se borra el contenido del bufer, por lo que la salida almacenada se pierde. Es decir, el `echo` anterior no se mostrará en la página.

El bufer, aunque recién limpiado, sigue activo. Por eso con el `echo "Otra vez escribo!!";`, el texto continuará insertándose en el bufer.

Luego con la línea `ob_end_clean()`; se borra el contenido del buffer y se deshabilita. Hemos perdido otra vez todo lo que se hubiera escrito en la página a partir del inicio de uso del buffer, así que el anterior `echo` no se mostrará.

En la siguiente línea hacemos un `echo "Esto si que va a aparecer en la página
";` Como habíamos deshabilitado anteriormente el buffer con `ob_end_clean()`, ese texto se irá directo a la salida y llegará al navegador que ha solicitado la página.

Luego en el código vemos una declaración de una función:

```
function convierte_caracteres_especiales($buffer){
    return htmlentities ($buffer);
}
```

Esta función se va a utilizar para ejecutarla antes de enviar el buffer al cliente. Recibe un parámetro que es el buffer que se está procesando. Dentro de la función se pueden realizar acciones y se debe devolver un valor, que será lo que definitivamente se envíe al navegador del visitante.

Para decirle a PHP que se debe ejecutar esa función antes de enviar el bufer al cliente se debe iniciar el uso del bufer `ob_start(parámetro)`, con el parámetro que es el nombre de la función, tal como se había comentado antes.

```
ob_start("convierte_caracteres_especiales");
```

El siguiente echo "Tenía que probar más cosas. Mañana espero que se lea con interés."; se coloca en el buffer. Y finalmente con `ob_end_flush()`; se envía el buffer al cliente.

Como al iniciar el bufer se había indicado un parámetro con el nombre de la función `convierte_caracteres_especiales()` se ejecutará esa función antes de enviar el contenido al cliente web.

En esa función simplemente se procesa el buffer, convirtiendo los caracteres especiales que tenga en sus correspondientes códigos especiales de HTML, con `htmlentities()`.

Por tanto, el texto que se enviará al cliente es el texto que haya en el buffer después de ejecutar `htmlentities()`.

Conclusión y resultado del control de salida de PHP

Para acabar, podemos ver el código que se generaría y se enviaría al cliente como resultado del procesamiento de esa página.

```
<html>
<head>
  <title>Página procesada con buffer de salida</title>
</head>
<body>
  Esto si que va a aparecer en la página<br>&lt;br>Tenía que probar más cosas. Mañana
  espero que se lea con interés.</body>
</html>
```

Artículo por [Miguel Angel Alvarez](#)

Mostrar código PHP de un archivo con colores resaltados

Vamos con un pequeño truco que nos ofrece una de las funciones de PHP, que sirve para mostrar código resaltado con colores. Es una función útil para mostrar en la salida un trozo de código PHP, pero con distintos colores y saltos de línea, lo que facilita su lectura.

PHP incorpora unos patrones para resaltar texto con código PHP, en distintos colores. Este se puede invocar con la función `highlight_file()`, que recibe dos parámetros:

- Primero, el nombre del archivo que queremos que muestre su código resaltado
- Segundo, un parámetro opcional que es un booleano, con valor por defecto `FALSE`. El valor verdadero o `TRUE` significa que queremos que nos devuelva un string con el código resaltado. El valor falso o `FALSE`, que es el comportamiento predeterminado, sirve para que nos muestre en la salida el código PHP formateado y con colores.

Esta función puede ser útil en páginas de tutoriales de PHP, para mostrar código resaltado con colores sin tener que complicarnos la vida, así como en la documentación de programas realizados con PHP.

Para ver la función en marcha es muy sencillo. Simplemente tenemos que llamarla así:

```
<?
```

```
highlight_file ("fichero.php");
?>
```

Se supone que "fichero.php" estará en el mismo directorio donde está el archivo PHP con el que estamos trabajando.

Si fichero.php tuviera un código como este:

```
<html>
<head>
  <title>Probando</title>
</head>

<body>

<h1>Hola amigos</h1>

<p>Esto es una <b>prueba</b> para ver como funciona</p>

<?
function tiene_acentos($cadena){
  $buscar = "áéíóúüÁÉÍÓÚÜñÑ";
  for ($i=0; $i<strlen($cadena); $i++){
    if (strpos($buscar, substr($cadena,$i,1))!==false)
      return true;
  }
  return false;
}
?>

</body>
</html>
```

El resultado que visualizaríamos, con el código resaltado, es este:

```
<html>
<head>
  <title>Probando</title>
</head>

<body>

<h1>Hola amigos</h1>

<p>Esto es una <b>prueba</b> para ver como funciona</p>

<?
function tiene_acentos($cadena){
  $buscar = "áéíóúüÁÉÍÓÚÜñÑ";
  for ($i=0; $i<strlen($cadena); $i++){
    if (strpos($buscar, substr($cadena,$i,1))!==false)
      return true;
  }
  return false;
}
?>
```

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

Artículo por *Juliana Monteiro Lazaro*

Obtener capacidades del navegador con PHP y `get_browser()`

En este artículo vamos a mostrar un mecanismo para averiguar las capacidades del navegador del usuario con PHP. Es decir, un método para saber si el navegador soporta Javascript, CSS, iframes y cosas similares. Es un método sencillo gracias a la función de PHP `get_browser()`, que nos devuelve un objeto donde podemos conocer fácilmente las capacidades del browser del visitante.

En cualquier momento con PHP podemos averiguar el navegador que está utilizando el usuario con la variable de sistema de servidor `$_SERVER['HTTP_USER_AGENT']`. Pero esta variable nos ofrece una información que no es del todo fácil de interpretar. Por ejemplo, en Internet Explorer 7, esa variable tendrá un valor como este:

Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1)

Si utilizamos Firefox en el mismo ordenador, al ver el contenido de esa variable obtendremos algo como esto:

Mozilla/5.0 (Windows; U; Windows NT 5.1; es-ES; rv:1.8.0.10) Gecko/20070216
Firefox/1.5.0.10

En realidad a partir de esa información y poniendo un poco de nuestra parte, podríamos deducir las funcionalidades que soporta el navegador. Pero vamos a ir un poco más allá y vamos a utilizar una función que nos va a devolver directamente la información bien trabajada y lista para utilizar. Directamente podremos saber si el navegador soporta distintas funcionalidades o tecnologías, sin necesidad de deducir nada y sin necesidad de conocer todos los navegadores existentes.

La función en concreto es `get_browser()` y es muy sencilla de utilizar. Nos devuelve un objeto que contiene toda la información que podamos necesitar sobre las capacidades del navegador. Para empezar, vamos a mostrar su utilización:

Nota: Existen algunos casos en los que no podremos utilizar directamente `get_browser()` y en los que necesitaríamos editar el archivo de configuraciones `php.ini` o descargar una

versión actualizada de la base de datos de navegadores y funcionalidades soportadas. Todo esto lo vamos a ver también en este artículo.

```
$navegador = get_browser();
```

Con esta línea hemos cargado en la variable \$navegador las capacidades del navegador. Luego podremos preguntar a esta variable-objeto acerca de las distintas funcionalidades que queramos saber si soporta el browser del usuario. Utilizaremos un código como este:

```
if ($navegador->iframes){  
    echo "<p>Permite el uso de IFRAMES</p>" . $navegador->iframes;  
}else{  
    echo "<p>NO permite IFRAMES</p>" . $navegador->iframes;  
}
```

//Si lo deseamos, podríamos sacar una lista de las propiedades del objeto \$navegador con un recorrido genérico a sus propiedades de objeto.

```
while (list($key, $val) = each($navegador)) {  
    echo "<br>$key => $val\n";  
}
```

Para el navegador Internet Explorer 7 todas las propiedades listadas junto con sus valores serían algo como esto:

```
browser_name_regex => ^mozilla/4\.0 (compatible; msie 7\.0; .*windows nt 5\.1\.*)\.*$  
browser_name_pattern => Mozilla/4.0 (compatible; MSIE 7.0; *Windows NT 5.1*)*  
parent => IE 7.0  
platform => WinXP  
browser => IE  
version => 7.0  
majorver => 7  
win32 => 1  
frames => 1  
iframes => 1  
tables => 1  
cookies => 1  
backgroundsounds => 1  
cdf => 1  
vbscript => 1  
javaapplets => 1  
javascript => 1  
activexcontrols => 1  
css => 2  
cssversion => 2  
supportscss => 1  
minorver => 0  
alpha =>  
beta =>
```

```
win16 =>
win64 =>
authenticcodeupdate =>
stripper =>
isbanned =>
wap =>
ismobiledevice =>
issyndicationreader =>
crawler =>
aol =>
aolversion => 0
netclr =>
clrversion => 0
```

Como podemos ver, sería muy sencillo obtener cualquier dato sobre el navegador y las tecnologías que soporta. Por ejemplo, para saber si soporta javascript accederíamos a la propiedad `$navegador->javascript`. Si vale 1 es que soporta Javascript y si el valor es 0 es que no lo soporta.

Igual no estamos acostumbrados a trabajar con objetos en PHP. Entonces en ese caso podemos solicitarle a la función `get_browser()` que nos devuelva un array de valores asociativos, de esta manera:

```
$navegador = get_browser(null, true);
```

En ese caso accederíamos a los distintos valores del array `$navegador` de esta manera:

```
if ($navegador["cookies"]){
    echo "<p>Permite el uso de cookies " . $navegador["cookies"];
}else{
    echo "<p>NO permite usar cookies " . $navegador["cookies"];
}
```

Parámetros de `get_browser()`

Como hemos visto `get_browser()` permite la entrada de dos parámetros, ambos opcionales.

El primero es el user agent (la cadena esa rara que especifica el navegador y la versión). Si no indicamos user agent, PHP utiliza el del navegador que ha solicitado la página. Pero podríamos indicarle otro user agent a la función para que nos informase sobre las capacidades de otro navegador que no tiene por qué ser el del visitante.

El segundo parámetro que recibe es un booleano, que si es true quiere decir que deseamos que devuelva un array. Si no se envía un valor o el valor enviado en ese segundo parámetro es false, entonces indica que se desea recibir un objeto.

Archivo `browscap.ini`, con base de datos de capacidades de los navegadores

Para que esta función realice el trabajo correctamente tenemos que disponer de un archivo que contiene la base de datos de navegadores y sus capacidades. Ese archivo se llama generalmente `browscap.ini` y debe estar guardado en nuestro sistema e indicado correctamente en la configuración de PHP. Además, en nuestro sistema debemos disponer de una versión actualizada de `browscap.ini`.

Se puede obtener una versión actualizada de browscap.ini gratuitamente en el sitio [Browser Capabilities Project](#)

Este fichero lo tenemos que guardar en nuestro ordenador y debemos especificar la ruta donde se encuentra en el archivo php.ini. Tenemos que editar el php.ini en las líneas adecuadas. Algo como esto:

```
[browscap]
browscap = C:\php\php_browscap.ini
```

Si no hemos indicado la ruta para acceder al browscap.ini nos saldrá un error como este:

```
Warning: get_browser(): browscap ini directive not set. in
c:\apache\www\tallerphp\get_browser.php on line 15
```

Utilizar `get_browser()` si no tenemos posibilidad de editar `php.ini`

Si estamos utilizando un alojamiento en un servidor donde no podemos editar el archivo de configuraciones `php.ini`, por ejemplo en un hosting compartido, también tenemos oportunidad de beneficiarnos de las capacidades de esta función.

Para ello tenemos que utilizar una función de PHP que nos provee un tercero, que dispone de un sistema útil para obtener las capacidades de un navegador. Esto está explicado en el sitio web [PHP standalone get_browser\(\) for Browscap](#).

Este sitio propone la utilización de una función nueva llamada `get_browser_local()` que sustituiría a la función nativa de PHP `get_browser()`. Este método lo único que requiere es que nosotros pongamos el archivo `browscap.ini` actualizado en un lugar donde se pueda leer desde páginas PHP.

Lo que tendríamos que hacer es descargar el [módulo php-local-browscap](#) de la página indicada anteriormente. Además, descargar el archivo `browscap.ini` del sitio [Browser Capabilities Project](#). Entonces podríamos utilizar la función `get_browser_local()` de manera similar a lo que haríamos con la función `get_browser()`.

```
require_once('php-local-browscap.php');
$navegador=get_browser_local();
```

Como se puede ver, primero tendremos que incluir el módulo con el código de la nueva función y luego llamarla como hacíamos anteriormente.

La función `get_browser_local()` tiene nuevos parámetros que se pueden utilizar, todos opcionalmente:

El primer parámetro es el `user_agent` (null por defecto). Este parámetro es igual que en la función `get_browser()`, para indicar otro user agent que no sea el del navegador del visitante.

El segundo parámetro es igual también que `get_browser()`. Sirve para indicar si queremos que nos devuelva un array o un objeto. Por defecto devuelve un objeto, pero si le pasamos el valor `true` devolvería un array.

El tercer parámetro es la ruta donde hemos colocado el archivo browscap.ini en nuestro sitio web. Por defecto se supone que el archivo está en el mismo directorio que el código PHP que se está ejecutando y que se llama browscap.ini. Es decir, el valor por defecto de este parámetro es './browscap.ini'.

El último parámetro es un booleano que indica si queremos que se guarde en memoria el archivo browscap.ini para acceder a él desde caché en sucesivas ocasiones. Por defecto el valor es false, por lo que no se guardaría en memoria.

Conclusión sobre get_browser()

Como se ha podido comprobar, este archivo es de bastante utilidad para conocer las funcionalidades o tecnologías que soporta un navegador. Hemos visto distintos casos de utilización y modos de usarlo en alojamientos normales, que no dejan editar el php.ini o donde no estamos seguros que el browscap.ini del sistema se encuentre actualizado.

Artículo por Miguel Angel Alvarez

Propagar el identificador de sesión de PHP por cookies o URL

Para guardar en la sesión informaciones independientes para cada usuario, PHP debe identificar la sesión de cada cliente que se conecta a la página. Al entrar un nuevo usuario en la página, PHP genera un identificador de sesión que es único y que será siempre el mismo durante toda su visita a la página. Las variables de sesión las almacena PHP internamente, asociadas al identificador de la sesión y debe asegurarse que cada cliente pueda memorizar el identificador de sesión durante toda su visita.

PHP dispone de un par de métodos para poder propagar el identificador de sesión en cada página que visita el cliente: en una cookie, o bien la propaga a través de la URL.

En una cookie

Es el método más cómodo y viene configurado por defecto. Nosotros no tenemos que hacer nada. PHP se encarga de guardar la información de la sesión por nosotros en una cookie en el navegador del usuario. El problema de este método es que no podemos estar seguros que todos los usuarios acepten cookies de sesión en sus navegadores.

En una variable pasada por parámetro en la URL

Consiste en enviar el identificador en todas las URL como parámetro, para recoger por método GET. Es un poco más engorroso porque en cada URL aparecerá el parámetro, algo como esto: pagina.php?PHPSESSID=8af7f938a4ab81aa6406e3d57ea41081 Este método tiene también sus ventajas e inconvenientes. Por un lado es más fiable que las cookies, porque pasar el identificador por la URL es posible siempre, independientemente del navegador cliente o su configuración. Pero por otro lado, enviar el PHPSESSID por la URL puede traer problemas, porque puede hacer público su identificador de sesión a otras personas o programas o porque pueda almacenarse en favoritos o historial urls con identificadores de sesión antiguos.

Pasar el identificador por una cookie

En la mayoría de los casos PHP intentará guardar el identificador de sesión en una cookie, aunque php.ini tiene algunas configuraciones para alterar este comportamiento por defecto.

session.use_cookies nos sirve para indicar si queremos que se envíe la cookie con el identificador de sesión. Por defecto está activado con el valor 1.

Nota: acuérdate de reiniciar Apache cuando hagas cambios en php.ini para que se hagan efectivos.

Pasar el identificador de sesión como parámetro en la URL

Podemos hacer una prueba para ver qué pasa con las variables de sesión cuando no se dispone de cookies. Veremos que no se memorizan las variables de sesión en las sucesivas páginas. Esto se puede solucionar pasando el identificador de sesión por la URL. En PHP sería algo como esto:

```
<a href="leer_sesion.php?=SID?&gt;"&gt;Paso variable de sesión por URL&lt;/a&gt;</pre
```

Es un fastidio tener que escribir ese dato en cada URL, pero PHP también dispone de mecanismos para que las URL se escriban automáticamente con los identificadores de sesión, de manera transparente para el desarrollador. Esto se puede indicar desde el archivo php.ini en la variable de configuración session.use_trans_sid.

Nota: según el sitio de PHP la alteración de las URLs automática para propagar el identificador de sesión sólo se puede hacer si se ha compilado PHP con la opción --enable-trans-sid. En mi configuración de PHP sobre Windows a mi sí que me permite usar session.use_trans_sid y se propagan los identificadores de sesión sin que tenga que hacer nada.

Artículo por ***Miguel Angel Alvarez***

Xajax: Ajax para PHP

Ajax es una tecnología que utiliza a su vez otra combinación de tecnologías, como XML y Javascript, para realizar peticiones de contenido o computación de servidor sin tener que recargar la página en la que está el usuario. Es una tecnología que permite una nueva gama de aplicaciones interactivas en la web, mucho más ricas y rápidas, dado que no precisamos recargar todo el contenido de una página para realizar peticiones al servidor.

Referencia: Puedes ver qué es ajax en este artículo: [Qué es AJAX](#)
Tenemos otros artículos que explican Ajax, puedes buscarlos con nuestro directorio, en la [categoría Ajax](#).

Si hemos intentado alguna vez trabajar con Ajax para programar una página web, habremos comprobado que la tarea se complica bastante, teniendo que realizar diferentes trozos de código en distintos lenguajes de programación y en distintos archivos. Todo esto puede provocar dolores de cabeza o páginas con códigos difíciles de entender y de mantener. Esta cuestión sin dudas es uno de los problemas que trae Ajax a los programadores, sobretodo a los que intentan dar sus primeros pasos. Pero herramientas como xajax pueden ayudarnos bastante.

En este artículo vamos a conocer xajax, una clase realizada con PHP que nos permite utilizar Ajax, combinado con PHP, para la creación de aplicaciones interactivas, de una manera muy simplificada. Con xajax podemos fácilmente ejecutar funciones PHP, que se ejecutan en el servidor, cuando el usuario realiza acciones en la página. Luego, los resultados de esas funciones PHP se producen en la misma página, sin que se tenga que recargarse.

Xajax es un producto Open Source gratuito y compatible con los navegadores más comunes, como Firefox, u otros navegadores basados en Mozilla, Internet Explorer, Opera, etc.

Podemos encontrarlo en <http://xajaxproject.org>

Hacer unas primeras pruebas para comenzar a conocer la herramienta nos llevará pocos minutos. Lo veremos a continuación.

Descargar e instalar xajax

Para probar xajax debemos descargar la última versión de la clase, que podemos descargar directamente desde su página web: <http://xajaxproject.org>

Obtendremos un archivo comprimido que debemos descomprimir en cualquier lugar de nuestro espacio de publicación. Por ejemplo, podemos crear un directorio llamado xajax donde podemos colocar todos los archivos del .zip descargado.

No hay que hacer especiales acciones para instalar xajax, simplemente descomprimirlo en cualquier servidor Apache o IIS que tenga compatibilidad con PHP 4.3.x o PHP 5.x, o superiores.

Una vez descargado podemos probar los ejemplos que vienen en el directorio examples, siempre a través de nuestro servidor web compatible con PHP, ya sea en local o en un servidor web remoto.

Atención al directorio donde finalmente metemos los archivos de xajax, pues luego tendremos que incluir archivos que hay en dicho directorio, para lo cual deberemos recordar la ruta relativa desde la página donde estemos al directorio donde está xajax.

Página simple con xajax y PHP

Veamos ahora como realizar una página que utilice xajax, para ejecutar una sencilla función PHP como respuesta a una acción del usuario. El ejemplo es relativamente sencillo, incluso lo podemos hacer en pocos pasos, como una receta. Luego se podrá complicar todo lo necesario para realizar acciones más complejas.

1) Incluir con PHP el archivo donde está la clase xajax

```
//incluimos la clase ajax
require ('xajax/xajax.inc.php');
```

2) Creamos una instancia de un objeto de la clase xajax

```
//instanciamos el objeto de la clase xajax
$xajax = new xajax();
```

3) Escribimos una función en PHP, que luego llamaremos con por medio de ajax

Esta función es todo lo complicado que se requiera. Realizará acciones del lado del servidor y por tanto puede acceder a bases de datos, abrir ficheros o cualquier cosa que se nos ocurra. Luego en la propia función realizaremos una instancia de un objeto AjaxResponse, que utilizaremos para mostrar resultados en la página.

```
function si_no($entrada){
    if ($entrada=="true"){
        $salida = "Marcado";
    }else{
        $salida = "No marcado";
    }
}
```

```
}  
  
//instanciamos el objeto para generar la respuesta con ajax  
$respuesta = new xajaxResponse();  
//escribimos en la capa con id="respuesta" el texto que aparece en $salida  
$respuesta->addAssign("respuesta","innerHTML",$salida);  
  
//tenemos que devolver la instanciación del objeto xajaxResponse  
return $respuesta;  
}
```

El objeto `xajaxResponse()` sirve para realizar acciones en la página sin tener que recargar el documento. Dispone de varios métodos o funciones, como por ejemplo `addAssign()` que sirve para asignar un valor a una propiedad de un elemento de la página. En este caso se asigna el valor contenido en la variable `$salida` al `innerHTML` de la capa "respuesta", con lo que se altera el texto contenido en esa capa.

4) Asociamos la función PHP al objeto xajax

```
//asociamos la función creada anteriormente al objeto xajax  
$xajax->registerFunction("si_no");
```

Esta asociación permitirá ejecutar la función PHP desde una llamada a una función Javascript.

5) Antes de enviar cualquier contenido a la página, tenemos que ejecutar un método del objeto xajax para procesar las peticiones del que puedan llegar a la página.

```
//El objeto xajax tiene que procesar cualquier petición  
$xajax->processRequests();
```

Insistimos, esta llamada al método se tiene que hacer antes de escribir ningún contenido dentro del código de la página, es decir, antes que llegue al cliente ningún carácter de código HTML.

6) Escribir el código javascript necesario para procesar las llamadas a ajax.

```
//En el <head> indicamos al objeto xajax se encargue de generar el javascript necesario  
$xajax->printJavascript("xajax/");
```

Lo ideal es hacer esta llamada al método `printJavascript()` dentro del `<head>` de la página.

Si nos fijamos, el método recibe un parámetro, que es la ruta relativa para acceder al directorio donde están los archivos xajax descomprimidos.

7) Podemos hacer llamadas a las funciones PHP en cualquier lugar del código, como respuesta a las acciones del usuario con javascript.

```
<input type="checkbox" name="si" value="1" onclick="xajax_si_no(document.formulario.si.checked)">
```

Como podemos ver, desde un elemento de la página, como en este caso una casilla de verificación, al cambiar su estado, se llama a una función javascript para ejecutar la función PHP escrita anteriormente. Es decir, al pulsar el checkbox se desencadena el evento `onchange` y con él se llama a la función `xajax_si_no()` enviándolo como parámetro el estado (chequeado o no) de la casilla de verificación.

Con esto es todo tenemos todo lo necesario para entender el ejemplo. Podemos [verlo en ejecución](#) en una página aparte.

Podemos ver el código del ejemplo completo a continuación, pero tener en cuenta que para que funcione tiene que disponer del código de la clase xajax, que en este caso debe estar en un subdirectorio que cuelgue del directorio donde está el archivo del ejemplo.

```
<?
//incluimos la clase ajax
require ('xajax/xajax.inc.php');

//instanciamos el objeto de la clase xajax
$xajax = new xajax();

function si_no($entrada){
    if ($entrada=="true"){
        $salida = "Marcado";
    }else{
        $salida = "No marcado";
    }

    //instanciamos el objeto para generar la respuesta con ajax
    $respuesta = new xajaxResponse();
    //escribimos en la capa con id="respuesta" el texto que aparece en $salida
    $respuesta->addAssign("respuesta","innerHTML",$salida);

    //tenemos que devolver la instanciación del objeto xajaxResponse
    return $respuesta;
}

//asociamos la función creada anteriormente al objeto xajax
$xajax->registerFunction("si_no");

//El objeto xajax tiene que procesar cualquier petición
$xajax->processRequests();
?>

<html>
<head>
    <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=ISO-8859-1">
    <title>Si / No en Ajax</title>
    <?
    //En el <head> indicamos al objeto xajax se encargue de generar el javascript necesario
    $xajax->printJavascript("xajax/");
    ?>
</head>

<body>
<div id="respuesta"></div>
<form name="formulario">
<input type="checkbox" name="si" value="1" onclick="xajax_si_no(document.formulario.si.checked)">
</form>

<script type="text/javascript">
    xajax_si_no(document.formulario.si.checked); //Llamando inicialmente a la función xajax_si_no inicializamos el
valor de la capa con la respuesta
</script>
</body>
</html>
```

Nota: Para ampliar la información de Xajax explorar el [manual de Ajax para PHP](#).

Nota: En estos ejemplos hemos utilizado la versión 0.2.5 de Xajax. Por favor, leer la [introducción al manual de Xajax](#), porque tiene algunos detalles sobre la versión y el trabajo con Xajax que conviene

saber, así como el enlace para la descarga de la librería Xajax 0.2.5 para que podáis poner en marcha vosotros mismos estos ejemplos en vuestros servidores.

Referencia: Hemos publicado un artículo sobre cómo actualizar este y otros scripts de este manual a la versión 0.5 de xajax: [Actualizar a Xajax 0.5](#).

Artículo por **Miguel Angel Alvarez**

Poner una captcha en PHP en tres pasos

Integrar un captcha a un formulario nunca ha sido tan fácil como utilizar reCAPTCHA, un servicio gratuito para implementar un captcha en cualquier sitio web.

Primero habría que decir lo que es un captcha y para qué sirve. Se trata de un sistema para comprobar que un formulario ha sido escrito por un humano y no una máquina. Sirve para evitar que robots, bots o máquinas envíen información a través de los formularios que tenemos en una web. Posiblemente ya sepas lo que es un captcha, pero si quieres más información accede a la FAQ: [Qué es una captcha?](#)

En este artículo vamos a mostrar una manera de implementar un captcha en nuestro sitio web PHP en muy pocos y sencillos pasos. Para ello nos vamos a ayudar de un servicio llamado reCAPTCHA, que ofrecen gratis en la web <http://recaptcha.net/>. Gracias a este servicio podremos olvidarnos de casi toda la complejidad de instalar y configurar la captcha en nuestro servidor y además contar con la seguridad de disponer de una captcha en continua revisión y actualización.

Paso 1: Registrarse en reCAPTCHA

Nos tenemos que dirigir a <http://recaptcha.net/> y registrarnos como usuarios. Nos solicitarán unos pocos datos personales para darnos de alta.

Una vez registrados debemos obtener unas claves para utilizar el sistema de captcha. Estas claves son específicas para cada dominio donde quieras utilizarlo. Las claves no son más que una llave pública y otra privada, que utilizaremos luego para poner la captcha en nuestra web.

Son algo como esto:

Public Key: 5LfCABCDEFHJIJUNDSBOejHq-5n5StSWawBpCAMX
Private Key: 6LfCAAfHJSDFGHHJHDeju3a-Z5lomjShHKaGh9g

Esas claves son, lógicamente, secretas y debemos mantenerlas así. Yo me he inventado estas claves para no mostrar las mías ;)

En el sitio de reCAPTCHA tendremos que descargarnos unos códigos PHP con una librería PHP para poner en nuestro servidor y unos ejemplos de uso del sistema. Esto lo podemos conseguir en el apartado "Resources". Veremos que tienen un plugin o librería para trabajar con PHP, que además está perfectamente documentado.

Una vez descargado el zip que contiene la librería PHP, la tenemos que poner en algún lugar de nuestro servidor. Recuerda luego donde la has puesto, porque tendrás que hacer un include

con PHP de esa librería.

Por ejemplo, podríamos colocar dicha librería en el mismo directorio donde está la página PHP que va a hacer uso de ella. Entonces la incluiríamos con algo como esto:

```
require_once('recaptchalib.php');
```

Paso 2: Incluir el captcha en el formulario

En la librería descargada ('recaptchalib.php') Hay una función que sirve para mostrar la captcha. Simplemente tenemos que llamarla con los parámetros correctos. La función devuelve el código HTML que tenemos que colocar en la página para que se vea la captcha.

```
recaptcha_get_html($captcha_publickey, $error_captcha);
```

Los parámetros que recibe son la llave pública que conseguimos anteriormente y un código de error, que es opcional. Luego veremos de dónde podría venir ese código de error.

El código del formulario sería algo como esto:

```
<?
require_once('recaptchalib.php');
//Llaves de la captcha
$captcha_publickey = "6Lfc?.";
$captcha_privatekey = "6Lfc? ";
//por ahora ponemos a null el error de la captcha
$error_captcha=null;
?>

<form action="miejemplo_formulario.php" method="post">
Nombre: <input type="text" name="nombre" size="30">
<br>
Edad: <input type="text" name="edad" size="3">
<br>
<?
//escribimos en la página lo que nos devuelve recaptcha_get_html()
echo recaptcha_get_html($captcha_publickey, $error_captcha);
?>
<br>
<input type="submit" value="Enviar">
</form>
```

Paso 3: Validar la captcha

Existe otra función para validar la captcha, llamada `recaptcha_check_answer()`. Esta función recibe también varios parámetros: La llave privada, la IP del usuario, y dos campos que contienen los valores que envía la captcha dentro del formulario `$_POST["recaptcha_challenge_field"]` y `$_POST["recaptcha_response_field"]`.

Esta función devuelve un objeto que tiene dos propiedades:

`is_valid`, un booleano para decir si se ha validado correctamente la captcha. `error`, un código de error que especifica qué ha ido mal si no se validó correctamente el texto.

Podríamos validar la captcha con algo como esto:

```
$captcha_respuesta = recaptcha_check_answer ($captcha_privatekey,
$_SERVER["REMOTE_ADDR"],
$_POST["recaptcha_challenge_field"],
$_POST["recaptcha_response_field"]);
if ($captcha_respuesta->is_valid) {
    //todo correcto
    //hacemos lo que se deba hacer una vez recibido el formulario válido
}else{
    //El código de validación de la imagen está mal escrito.
    $error_captcha = $captcha_respuesta->error;
}
```

Llamamos a la función `recaptcha_check_answer()` enviando los parámetros comentados y guardamos el valor devuelto por la función en la variable `$captcha_respuesta`.

Luego comprobamos si el atributo `$captcha_respuesta->is_valid` es true. En ese caso sabemos que el texto escrito de la imagen se ha validado correctamente y hacemos lo que haya que hacer en el formulario.

Si `$captcha_respuesta->is_valid` era falso, entonces quiere decir que no estaba bien escrito el texto de la imagen. Podemos entonces actualizar la variable `$error_captcha` para que cuando mostremos de nuevo la captcha podamos pasarle el error generado y que avise al usuario.

Tenemos nuestra captcha funcionando!

El código completo de este ejemplo es el siguiente:

```
<html>
<head>
    <title>Página con formulario protegido por captcha</title>
</head>

<body>
<?
require_once('recaptchalib.php');
//Llaves de la captcha
$captcha_publickey = "6Lfc?.";
$captcha_privatekey = "6Lfc? ";
$error_captcha=null;

if ($_POST){
    $captcha_respuesta = recaptcha_check_answer ($captcha_privatekey,
$_SERVER["REMOTE_ADDR"],
$_POST["recaptcha_challenge_field"],
$_POST["recaptcha_response_field"]);
    if ($captcha_respuesta->is_valid) {
        //todo correcto
        //hacemos lo que se deba hacer una vez recibido el formulario válido
        echo "Todo correcto!";
    }else{
        //El código de validación de la imagen está mal escrito.
        echo "Has escrito mal el texto";
        $error_captcha = $captcha_respuesta->error;
    }
}
?>

<form action="miejemplo_formulario.php" method="post">
Nombre: <input type="text" name="nombre" size="30">
```

```
<br>
Edad: <input type="text" name="edad" size="3">
<br>
<?
//escribimos en la página lo que nos devuelve recaptcha_get_html()
echo recaptcha_get_html($captcha_publickey, $error_captcha);
?>
<br>
<input type="submit" value="Enviar">
</form>

</body>
</html>
```

El ejemplo se puede [ver en una página aparte](#).

Conclusión

Hemos visto en 3 pasos como hacer un formulario seguro con un captcha, apoyándonos en la tecnología desarrollada por reCAPTCHA. Hemos utilizado PHP para el ejemplo, y lo hemos simplificado al máximo. Aunque hay que señalar que este sistema dispone de un API para poder implementar captchas en otros entornos, incluso con Javascript, y poder personalizar el aspecto del cuadro que solicita introducir el texto de la imagen. También disponen de plugins para integrar reCAPTCHA en aplicaciones web populares como phpBB, MediaWiki o WordPress.

Artículo por *[Miguel Angel Alvarez](#)*

Redirección PHP 301 y 302

Una redirección sirve para llevar al navegador del usuario a una página distinta. Redirigir al navegador nos puede servir para enviarlo a otra dirección URL distinta donde están los contenidos que desea ver.

Existen dos tipos de redirecciones, la 301 que quiere decir "Redirección permanente" y la 302 que significa "Redirección temporal". El usuario que nos visita no percibe si estamos haciendo una redirección de un tipo u otro por PHP, pero el tipo de redirección utilizada si resulta una información interesante para buscadores, porque entenderán que una dirección ha cambiado temporal o permanentemente y eso les servirá para tener actualizadas sus bases de datos. Esto se explica en la FAQ: [Diferencias entre redirección 301 y 302](#).

Con PHP podemos redirigir al navegador con la función `header()`, que envía informaciones en la cabecera del HTTP.

Una redirección con PHP se haría con algo como esto:

```
header("Location: http://www.una-redireccion-cualquiera.com");
```

Por defecto PHP realiza una redirección temporal, de tipo 302. Pero nosotros podemos indicarle otro tipo de redirección, también con la función `header()`, indicando el tipo antes de hacer el `Location`.

Para hacer una redirección 301 (permanente), utilizaremos un código PHP como este:

```
header("HTTP/1.1 301 Moved Permanently");  
header("Location: nueva_pagina.html");
```

Para hacer una redirección 302 con PHP (temporal) el código sería así:

```
header("HTTP/1.1 302 Moved Temporarily");  
header("Location: nueva_pagina.html");
```

Recordar entonces que es bueno de cara a buscadores que una redirección temporal, tipo 302, en algún momento la cambiemos por una de tipo permanente (301), si es que los contenidos que se han movido siempre van a mantener esa localización.

*Artículo por **Miguel Angel Alvarez***

Números aleatorios en PHP

PHP dispone de una serie de funciones para generar números aleatorios fácilmente. Las vamos a ver explicando sus usos y diferencias de funcionamiento. La forma más básica de generar un número aleatorio en PHP consiste en dos pasos:

```
//alimentamos el generador de aleatorios  
srand (time());  
//generamos un número aleatorio  
$numero_aleatorio = rand(1,100);
```

Como vemos, en el primer paso se utiliza la función `srand()` para alimentar la semilla de generación de números aleatorios. Este paso es necesario sólo en versiones anteriores a PHP 4.2.0, pues a partir de esta versión este paso se hace automáticamente. A la función `srand()` hace falta enviarle un valor para alimentar la semilla. Nosotros enviamos lo que devuelve `time()`, que es un timestamp con el número de segundos desde el inicio de 1970.

Luego generamos un número aleatorio con la función `rand()` que recibe un par de valores opcionalmente, que son el mínimo y el máximo de los números aleatorios generados. En el caso anterior se consigue un número aleatorio entre 1 y 100, incluyendo estos dos valores entre los posibles.

Si no se indica nada a `rand()`, el valor mínimo será cero. El valor máximo depende de la plataforma donde se esté ejecutando PHP, por ejemplo en Windows el valor máximo sería 32786. Si queremos asegurarnos que este valor máximo sea mayor, entonces conviene definir los valores máximo y mínimo al llamar a la función.

Generación de números aleatorios con `mt_rand()`

PHP tiene otras funciones para generar los números aleatorios, aparte de las que hemos visto, que utilizan unos algoritmos mejorados para conseguir números al azar.

La función de PHP `mt_rand()` genera aleatorios con un algoritmo que es de promedio 4 veces

más rápido que el algoritmo que utiliza rand()).

El uso de mt_rand() es similar:

```
//alimentamos el generador de aleatorios
mt_srand (time());
//generamos un número aleatorio
$numero_aleatorio = mt_rand(0,5);
```

Primero se debe utilizar mt_srand() para empezar la generación de números aleatorios con una semilla. Pero este paso a partir de PHP 4.2.0 no es necesario, porque se hace automáticamente.

Luego se generan los números aleatorios con mt_rand(), a la que le pasamos el rango de valores que queremos obtener, con los parámetros mínimo y máximo. En nuestro ejemplo obtendremos valores aleatorios entre el 0 y el 5.

La generación de números aleatorios en PHP es muy sencilla, como se ha podido comprobar. Si queremos que los números aleatorios tengan decimales podemos probar el truco de la FAQ: [Números aleatorios decimales en PHP](#)

Artículo por **Miguel Angel Alvarez**

Código fuente creación de SMS Web

Basicamente se necesitan tener los "@dominio" pertinente a cada empresa de telefonía para envío de sms. Por ejemplo para movistar es numero-celular-sin-cero-ni-quinque@sms.movistar.net.ar El código html/php es trivial y variado, pueden hacerlo a gusto, igualmente acá envío mi ejemplo:

Lo primero que hacemos es crear un formulario en HTML con lo siguiente:

1. un formulario (form method="post" action="sms.php", etc...)
2. un campo de texto para ingresar el e-mail del remitente. (que contendrá el valor name="numorigen")
3. un campo de texto para ingresar el telefono celular del destinatario. (que contendrá el valor name="numdestino")
4. un campo select para escoger la empresa de telefonía que utiliza el destinatario. (contendrá el valor name="nomemp" y en cada valor del select (Value) contendrá la abreviacion dada para cada empresa como figura en el código value="xxx"... , (personal,ctid,ctig,etc)
5. por último: un campo para escribir texto (textarea) con el valor "mensaje" y un botón enviar.

El html va a gusto de cada uno. Ahora vamos a ver como queda el script php al que le pasamos los datos del formulario.

```
----- SMS.PHP -----
```

```
<?php
```

```
$varnumorigen = $_POST['numorigen']; // e-mail del remitente tomado desde el form.
$varnomemp = $_POST['nomemp']; // empresa de telefonía (destino) - idem.
$varnumdestino = $_POST['numdestino']; // numero de celular (destino) - idem.

// procesamos el select del formulario html, con switch desde php.

switch($varnomemp)
{
case "personal": // personal
$empresa = "@personal-net.com.ar"; // "case" valor "personal" - lo asigno a $empresa.
break;
case "ctid": // cti digital
$empresa = "@infotext.cti.com.ar";
break;
case "ctig": // cti gsm
$empresa = "@sms.ctimovil.com.ar";
break;
case "movistaru": // movistar ex-unifon
$empresa = "@e-mocion.net.ar";
break;
case "movistarm": // movistar ex movicom
$empresa = "@movimensaje.com.ar";
break;
case "movistarg": // movistar genérico
$empresa = "@sms.movistar.net.ar";
break;
case "nextel":
$empresa = "@nextel.net.ar";
break;
case "skytel":
$empresa = "@skytel.com.ar";
break;
case "conectel":
$empresa = "@conectel.com.ar";
break;
default:
echo "empresa incorrecta - seleccione nuevamente";
break;
}
$title = "sms web"; // titulo que aparecerá en el sms del destinatario
$headers = "From: " . $_POST['nombre'];
$headers .= "<" . $varnumorigen . ">\r\n"; // e-mail del remitente (esto es 100% obligatorio)
$headers .= "Reply-To: " . $varnumorigen; // esta campo no es obligatorio, pero queda bien :)
$message = $_POST['mensaje']; // esta variable contiene el mensaje que enviamos, captado desde el formulario
$destino = $varnumdestino.$empresa; // concateno el numero de celular con la empresa

mail($destino,$title,$mensaje,$headers); // enviamos el mail/sms !

echo "sms enviado!";

?>

---- SMS.PHP -----
```

*Artículo por **Jorge Jesús Churruca shock dude***


```
document.getElementsByTagName('body')[0].removeChild(oldsc);
sc=document.createElement('script');
sc.id="old_sc";
sc.src=url+'&'+Math.random();
document.getElementsByTagName('body')[0].appendChild(sc);
}
function addCss(cssCode,i) {
control=document.getElementById(i)
if(control)
document.getElementsByTagName("head")[0].removeChild(control)
var styleElement = document.createElement("style");
styleElement.type = "text/css";
if (styleElement.styleSheet) {
styleElement.styleSheet.cssText = cssCode;
} else {
styleElement.appendChild(document.createTextNode(cssCode))
}
styleElement.id =i;
document.getElementsByTagName("head")[0].appendChild(styleElement);
}
window.onload=function(){
enlaces=document.getElementsByTagName('a');
for(i=0;i<enlaces.length;i++){
cssCode='a#'+enlaces[i].id+':visited{font-family:Verdana, Arial, Helvetica, sans-serif;font-size:9px;color:#FF0000;background:url(<?php echo basename($_SERVER['PHP_SELF']) ?>?)'+enlaces[i].href+'}';
addCss(cssCode,'tt'+i)
}
}
</script>
</head>

<body>
<p><a id="uno" href="http://www.google.es">Google</a><br />
<a id="dos" href="http://www.yahoo.com">Yahoo</a><br />
<a id="tres" href="http://www.php-hispano.net">php-hispano</a><br />
<a id="cuatro" href="http://www.neocreativo.es">neocreativo</a></p>
<a href="javascript:rpc('?ver_en_historial')">ver cuáles figuran en tu historial</a>
<div id="lista"></div>
</body>
</html>
```

Lo complicado de nuestro experimento puede ser el javascript utilizado, pero podríamos haberlo obviado, ya que sólo lo usamos para no escribir manualmente los estilos de cada enlace. De todas maneras, vamos a explicarlo un poco:

La función addCss lo que hace es agregar vía DOM un elemento style en la página, con el texto que nosotros necesitamos que figure en él. Maneja un poco de cross-browser, ya que Explorer, a diferencia de los navegadores serios, no interpreta correctamente el texto agregado como TextNode dentro de un elemento style.

Explicada la función addCss, y como la función rpc sólo sirve para obtener del servidor el informe de los enlaces visitados de manera asíncrona, y ya fue explicada aquí, sólo resta decir que, cuando carga la página (es decir, se produce el evento window.onload), se exploran los enlaces y se les asigna a cada uno de ellos un estilo para su estado visited, que pone nuestra imagen php como background y al mismo tiempo le pasa por la url la ruta que deberá almacenar para tenerla disponible cuando solicitemos el informe de páginas ya visitadas.

Final paranoico / dramático

Si con un poco de css y php es posible acceder a datos supuestamente privados del usuario, qué otra información privada estaremos regalando sin saberlo...

Artículo por **Andrés Fernández**

¿PDF Mejorado?

Pero, mientras lo hacía, no dejaba de pensar en las veces que tuve que explicarle a uno que otro cliente que para leer los informes en PDF debía bajarse el Acrobat Reader, y me dije: "tiene que haber algo mejor"...

Segundos más tarde, Google me llevaba a [unipage](http://unipage.com) y a algunos sitios que hablaban de pdf mejorado... Luego, fue cuestión de estudiar el código fuente de unipage, traducir artículos de wikipedia y otros lugares, de consultar a un par de foros amigos y sacar variadas e interesantes conclusiones que compartiré con ustedes.

La primera sorpresa con la que me encontré fue la existencia de una manera de incrustar archivos externos (imágenes, swf, estilos, códigos javascript, etc.) diferente de la que yo conocía y manejaba y, para colmo, estandar: el esquema data: URI, que permite la inclusión de pequeños elementos de datos en línea, como si fueran referenciados hacia una fuente externa, y cuyo formato es el siguiente:

data:[<MIME-type>][;base64],<datos>

Donde ;base64 es la codificación.

Veamos un ejemplo de cómo mostraríamos una imagen con el esquema data: URI:

```

```

Hasta aquí todo bien, utilizando este esquema podía generar contenidos en un html sin depender de una conexión a internet ni de imágenes externas, sólo tenía que codificar la imagen en base64 con la ayuda de php* (ver cómo al final de la página) y luego generar un htm o un html descargable, todo perfecto y completamente standalone... pero había un problema... y bastante serio: El fucking Internet Explorer no soporta el esquema data: URI!

Cuando me di cuenta de esto, tomé el siguiente vuelo Google disponible, visité a varios amigos gurúes del Nuevo y El Viejo Continente y, lamentablemente, en todos lados me encontré con la misma respuesta que, ya antes que ellos, me había suministrado el Maestro Po cuando le planteé mis tribulaciones: "Si quieres que funcione en Intelnet Explolol, Pequeño Saltamontes, debes usar, en lugar del esquema Data: ULI, el formato MIME HTML".

No podía creerlo. Mientras miraba los muertos ojos del anciano (como diría Faulkner: semejantes a dos coágulos de flema), mi cerebro funcionaba a toda prisa: "Pero Maestro, salvo el cada vez menos repugnante Ópera, el resto de los navegadores no soporta el formato mhtml", le dije. El maestro hizo un gesto cansino. Finalmente dijo: "En efecto, si quieres que

CWAXBY7mZVm87dR1kNNugRQ6qCPboONSnuPsOgJ+PcWt8pjjv764mtvWmMfmluiSlti1clXIPy2/HQd5OL2KTW265GWT
1ZH3
yrGB6UCUXcy+JdnIRV+egOMjmZEx94mP/IXyRSPdzkVerkzfRxE/Qu0glvqavu0ABYWO2mjt+QY1PSsbtftQnhWIIrhOoII9
+gcOJz2X5ZhrfM56/nyWQkg2zTzNuKwRhYVgUGiqrAr5AB79BIVu8HIOA2uPxOUxjZjjF/E9zjZGla2ubOTcFnEDGMHjYD
eqsKA1poA/uPz3JctvjxHjxaxxDyqgiacySZCOUb4jLKAigtD9C9CdAqrm1u7F/RvbeW2mArsnjaNto6BqMBUH2EaC7w/HL
j+lyuXtm/tsm+a3tG/5LpYh7AfpUtQddAS2DBMBiZiWjW1kZSqnxQ0Hh8TQ6CdLbRf3C3wUDF7gu6XMIvosqVUVV+EjgfnT
BHyXqrkTGaOnoFLiUUB27OkjAfu203DQC2KAbAZmK7bZjC0fpTkVAnDhBqEhNT8tAZYK6ysHdCvc2tiLmWONUsrOR1Wovv
UYK
7uXUBPbrjoA3kcWcx+4Hks7PnWlt3owcBAQ/khqNhjCny7PL08dA0u2nuJfWmJxb8D4OuQ7zvG1nkEy2QSLBIkVNI7bC7IV
D dSkrSMMdrA7BoFbyibk78zyM/JLcR8je633VrdTChJ8qAuxBjHvDfroc/NnKcZSLibNCRP6e23hCWFPKSSnIHUOOp7IZYZ
ibaid7UjKmdSPRMO3L1A8oevh0+Ogs7ZXEd1KrM2cbIENDTzqBkISSCelSdg8p0H//Z
---=boundary-1

Content-Type: image/jpeg

Content-ID: <pepe2>

Content-Transfer-Encoding: base64

Content-Disposition: inline; filename="red-test-image.jpg"

/9j/4AAQSkZJRgABAQAAZABKAAAD/7AARRHVja3kAAQAEAAAAMQAA/+4AUIFkb2JlAGTAAQAAAAQMAEAMDBGkAAAQIAAA
H+AAACvD/2wCEAAAGBgyYGBggGBggMCAcIDA4LCAgLDhANDQ4NDRARDA4ODg4MERATFBUUEAZGRsbGRkIJCQkJSkpK
S

kpKSkpKSkBCQgICQoJCwoKCw4MDgwOEg4ODg4SFA4ODw4OFBoSEBAQEBIaFvgVFRUYFxcwGhocHcmjIiMjKSkpKSkpK
SkpKf/CABEIAFAAUAMBIGACEQEDEQH/xAC6AAADAQACAwAAAAAAAAAAAAAGBwgFAwQAAQIBAQAAAAAAAAAAAAAAAAAA
AA

AAAQAAICAgEDBAMAAAAAAAAAAAAQFAQMcbGAREhMQQBYHiiMVEQACAQMCAwUFBgQHAAAAAAAAABAgMAEQQHfBE1Fh
Ij

IFEHGBkRShscFSYjNCoimVIHKSWMkRBIBAAAAAAAAAAAAAAAAAAAAQBMBAQACAgICAgMBAAAAAAAAAREAITFBUWFxg
ZGh

QLHB0f/aAAWDAQACEQMRAAAafnz66JnodbdMnt4HxSqMBEW4WH4HF56XLCCiY3cLMI4MvKzQhw+h18D1bxfSp2UA+5U
HPu/TiAxdMoWFmOUzPoPU9NzICKc6kQwQONkdEHAtWSvjvpxaxznYa6a4x5jrRciMIhDRHoLqwn5PWlmH1Uak2z//2g
AIAQIAAQUb9z//2gAIAQMAAQUb9z//2gAIAQEAAQUb5142dAJBjN+clTnsex2QJub8etPt61IXjMZY+ppdS8M869mZ

3xwF4QAFXn2x3xlxczYKrEGwDvaPTEtmXnFLXCGHIf10vwwn6/1/KCfrcOeMtPeKuVZ2TXK4ySsab6SqI5uMifHtRN
GCujeFNeCt+sc4sNvSLb/n6ictmgE4yuevNjIm9Cy2TjFdt4yeEactuIap0u4u1NrwCUr+OPsTtjrBiyPGUJEYeLmr

BSvU7ZXdlUpJoi+v74wDjhLzIUUpQ9HsbN7qRK37KBJOWONTGuPmI9uzvFpWuV98TqRCNjTxl2OWwraiq9ogJmzYLm
1Nsdsc/16Vn1e7AZSoYomAzVsFfI1FNTPMmhyLB9T2Srcp8uc9cunbXoAWVhWxKDGtBIVoVkiMRXp6wifwM4ab

pcVvJzPbHXObcpwnV5vzVft1oPGRywhJhE5RFvYNE/nPxmNfjowiIn4aRerppwop/9oACAECAGY/AU//2gAIAQMcbj
8BT//aAAgBAQEGPwH2fUzr6n9uFdXc/pH41bCWPCj5G3Vf4Ixt/loN/cJShNtyqF1428C1t6sOZ+qZNR8Yyt/jQTLd

cPLGjxyGyHvRzofdxoMhDkeBGo/wT5037cCFyBxNuQ99T2ZUSZJmJHYo5KL8hwFG1ZOAKSNDI/uOd4k4bbqysAPI7/
ZbiOYoSenzsgU3aA3MTdzJwpiB0cuH9+A8v1Kean2yKi7vqnWfj+UG7k/y2qPAwIDSvqWY2VQOJYDnZcsvgPpWRR3

ahiasHyF7w6/ihq+HnyR90qCT7V6f3UZkj+qhHGSC7ED9Seb7K++k9QCdRbdOdOZjYi9u8WqPjx23wzKHjbtB1Hsyx
lm26wgtxMt7pa/2916yJ5Ttgw4DJJwF2YkhhCg2Tj5MG7yXj4jvtepTgOWeHV4yNre8A0cWdpGIXzqi329xvbWIMWN

klb/ANSSwG0fBjeo/UcBhbKuNyCwZhzI7asaWI/+WV4Qe0A7x9jvII/TMqVxcRyFAkJtYbi5N1XXiRVsoqsMB8GNH5
A3C5P8RHb8q9SyygTjOyXRB9Vug36/Eio8r1/LfImV7yqDuR4/4Y18pTXjapZcJHHVWxLtuNr7rDuFzU76ik0QD2Xp

flQdMBRjThQjizmdyTYkWLZ2Gg5ioospCsatvT8RPLaaXFLATdy5eXS3iezcbkMLW1FT+oRMjLHE0cEoJRkkIBtY
+YG1qHGxrMwj5kl6g9zAA/d7J4/7dkywwW3ZUY3Lf54sNdL1J5EU+HkZJO6NwO

mWLWS1tQeRq7WY8TuOtu6l/wCOMbE/yiomPmQbZB3DgaiN5E5k03aiXs0aqdbaG4F6mx+sPrMjan0YN5UkD
gsjoNRttY3rp8COIPIjkaJbySJZ/iQPxpDPcmQ7Yo1F2bmfISZWZnx46GEbcQHe4J8V5AgNja2lZeRD0s7FyJC8Rtl

jvppcUnqCSsvmQY7aEHm7d3ZRN9O3kaWwAIBB79bipI21V8dvgysHBrB9TkW1kx7suZlImVlGLFdrX28ayM3Fh6UTW
UH89rjQeAWLUs6cgQR3GvTvU5239OCRfnpUePqW6G3ZYVO/RPTMEY1j/AE3G039yIRrS5Xqvpmp4QNVtQMq257rmpAv

Akwm4a1a2lbu37qvVTPcI9CLvYkO3ysPnSj07j+mnGkm6+yRPYuBe9jw0owI45WDMNOV9vgO0tqBpu0HfX+2sCRW68
MCbYlbnEfNhfU6V/WgeM87eL5cKaHHiCxtL8bU8TqT20PxorLyte7jqK62RCsBnkZ0RBYFTba1hWuBX/YyVDco1

8T/AOlbmvUoPTsI6C7XAW53B76EnQ62oG1jX08kKuu/erm+5SbAgd2IEAVoL0X2at4deQqNeRYDjt4m3m5VHi5bwtl
7zuyCXcxQk7tsd7biLc7cajx4tI4ICIO5RYV//9oACAECAGY/AU//2gAIAQMcbj

6kgnSp2tHnHLwD2/0jC6cSKSXIDZ4xIMFBRKaP245qMMFp4UH043yjlGrEHsTnLlxYzBNWk2jS2qtGC8Aqq31Q
4UAINhIw8eMunywSsRfPIpRh+Q0QwLKT8sw87E/KoeR2RONI8UVXbj/ua6elHELcJrThQcn8XzluTKjVEHR6FzTiq

L7sa95p8YYQe6/U/WiwociX5Y/LASDXvIqAOKog84AJ8o4+5ivFzTIFbKBLhjQBpHSMT4dmXU684hoIkMofcnDA/U0
tcM285tgB2jxYFPKeLxyAusOifku9HpgGrBDI7ZovGsXAuakWo6895el2iw61m+3q+I/wBR+sY1tOEygnJ/eCFJ

bNkk7FrpzoOf5HwKRYIWGnkiliA6OGE5xqBa4MRQKHLXA5xCieQeW6YINn5hiQDlcW45odgkdY8CEUYo75zSxiOCiSB
g1OMarts0UK71cqLUOIm+tZrBge38NIzpuGP4mAXQw8V4HzxvBiwwYE3MrTvwxA2Aiujuv8Y8AI14OjrEivnVSN9Y+F

EvhKwmcZxOjICY3Iw2acBeuTEGrYaOhB3zjaqb2bX6RiFN6KBoPA2rowuwiLm+kY5ZfeL1Oibs58uP6zSYtbdacOck
uxYOC+OcvDkHvYj8YRK9jDL0m44f6LF2PkWtj8gJ9WhrbORAIp93g5N9LFX2YFHXFdSjo8YNDk+Vdxw3vBKW9acf

LhSADIE614zUZHf/v1kuOYXUNtv6OIFk5qj6lgbLfcFkQ2UuXzjKk7rUE7um7CYIbqG1qjjB0QeR9AbGa+fXE426d
sr/cJaCTX5zY2z/sYQHGujsrN5LoWIEI4aGesKksGaATACB1nz9VZda/rZkqqrZwvndunpxhoEb7J1I553Aep4qHjOSBO

X+8q/QP8wUqUCAIcZb3lxDsU9G0Mb5msGelha6epDQVzDC0r9tCbvgz//Z
---=boundary-1--


```
AAAAAABAgMAEQqHEjFBE1FhIjIFEHGBkRShscFSYjNCoiMVIHKSwiMkRBIBAAAAAAAAAAAAAAAAAAAAQBMBQAQACAgICA
gm
BAAAAAAAAAREAITFBUWFxgZGhQLHB0f/aAAwDAQACEQMRAAAafnz66JnodbdMnt4HxSqmbEW4WH4HF56XLCCiY3cLMI
4MV
kZQhw+h18D1bxfSp2UA+5UHPu/TiAxdMoWFmOUzPoPU9NzICKc6kQwQ0NkdEHAtWSvjvpzaxznYa6a4x5jrRciMIhDRHoL
qmw5PWlmH1UAK2z//2gAIAQIAAQUB9z//2gAIAQMAAQUB9z//2gAIAQEAAQUB5142dAJBjN+cITnsex2QJub8etPt61IXj
MZY+ppdS8M869mZ3xwF4QAFXn2x3xlxczYKREgWdvaPTeTMxNfXLCGhIf10vwwn6/1/KCfrcOeMtPeKuV2ZTxK4ySsab6
SqI5uMifHtRNGCujeFNeCt+sc4sNvSLb/n6ictmgE4yuevNJIIm9CY2TJfDt4yeEactuIAp0u4u1NrwCUr+OPsTtjrBiyP
GUJYELeMrBSvU7ZXdUurjOI+v74wDjhLzIUUpQ9HsbN7qRK37KWJOWONTGuPmI9uzvFpWuV98TqRcCNjTxl2OWwraiq9og
JmzYlM1Nsdscsc/16Vn1e7AZSoYOmAzVsFfI1FNTPMmhyLB9T2Srpc8uc9cunbXoAWVhWxKDGtBIVolvkiMRXp6winfw
M4abpcVVzJPbHXObcpwnV5vzVfT1oPGRywhJhE5RFvYNE/nPXmNfjowiIn4aRerppwop/9oACAECAGY/AU//2gAIAQMcb
j8BT//aAAGBAQEGPwH2fUZr6n9uFdXc/pH41bCWPCj5G3Vf4xt/loN/cJShNtyqF1428C1t6sOZ+qZNR8Yyt/jQTLdc
PLGjxyGyHvRzofdxoMhDKeBGo/wT5037cCFyBxNuQ99TZ2USZJmJHYo5KL8hwFG1ZOAKSNDI/uO4k4bbqysAPI7/Zbi
OYoSenzsgU3aA3MTdzJwpiB0cuH9+A8v1Kean2yKi7vqnWfj+UG7k/y2qPAwIDSvqWY2VQOJY0DnZcsvgPpWRR3ahias
HyF7w6/ihq+HnyR90qCT7V6f3UZkj+qhHGSC7ED9Seb7K++k9QCdRbdOdOZjYi9u8WqPjx23wzKHjbtB1Hsyxlm26wgt
xMt7pa/2916yJ5Ttgw4DJiWf2YkHbCg2Tj5MG7yXj4jtvepTgOWeHV4yNre8A0cWdpGIXzqi329xvbWIMWNklb/ANSSw
G0fBjeo/UcBhbKuNyCwZhzI7asaWI/+wV4Qe0A7x9jVII/TMqVxcRyFAkJtYbi5N1XXiRVsoqsMB8GNH5A3C5P8RHb8q
9SyYgjTOyxRB9Vug36/Eio8r1/LfImV7yqDuR4/4Y18pTXjapZcJHHVWxLtuNr7rDuFZuT6ik0QD2XpflQMBRjThQj
ZmdyTykWZL2Gg5ioospCsatvVT8RpRLaaXFLATdy5eXS3iezcbkMLW1FT+oRMjL
HE0c0EoJrkkIBtY+YG1qHGxrMwj5kl6g9zAA/d7J4/7dkywwW3ZUY3Lfs4sNdL1Jj5
EU+HkZJO6NwOmWLWS1tQeRq7WY8TuOtu6l/wCOMbE/yiomPmQbZB3Dgai9NSE
5k03aiXs0aqdbaG4F6mx+sPrMjan0YN5UKDgsjoNrttY3rp8COIPIjkaJbySJZ/iQPxpDPcmQ7Yo1F2bmfISZWZnx46G
EbcQHe4J8V5AgNja2lZeRD0s7FyJC8RttljvppcUnqCSsvmQY7aEHm7d3ZRN9O3kaWwAIBB79bipI21V8dvgysBrB9T
kw1kx7suZlIimVLgLFdrX28ayM3Fh6UTWUH89rjQeAWLUs6cgQR3GvTvU5239OCRfnpUePqWG63ZYV0/RPTMEY1j/AE3
G039yIRs5Xqvpmp4QNvtQMq257rmpAvAkmw4a1a2lbu37qyvVTpCi9CLvYk03ysPnSj07J+mnGkm6+yRPyuBe9jw0ow
l45WDMN0V9vg00tqBpu0HfX+2sKRW68MCbYlbnEfNHftU6V/WgeM87eL5cKaHHiCxtL8bUb8TqT20PxorLytde7jqK62
RCsBnkZORBYFTba1hhuwBX/YyVDco18T/AOlbmvUoPTsI6C7XAW53B76EnQ62oG1jX08kKuu/erm+5SbAgd2IEAWoL0X2
at4deQqNeRYDjt4m3m5VHi5bwtl7zuyCXcxQk7tsd7biLc7cajx4tI4ICIO5RYV//9oACAECaWE/EP5P/9oACAEDAwE
/EP5P/9oACAEBaWE/ElvHaHOak6kgnSpx2tHnHXLwD2/0jC6cSKSXXIDZ4xIMFBRKaP245qMMFp4UH043yjlGrEHsT
nLixYzBNWK2jS2qtGC8Aqq31Q4UAINhIw8eMunywSsRoFPipRh+Q0QwLKT8sw87E/KoeR2RonI8UVXbj/ua6elHELc
JrThQcn8XzluTKjVEHR6FzTiqL7sa95p8YYQe6/U/Wiwoxi5Y/LASDXvIqAOKog84AJ8o4+5ivFZtIFbKBLhjQBpHS
MT4dmXU684hoIkMofcnDA/U0tcM285tgB2jxIYFPKeLxyAusOi3fku9HpgGrBDI7ZovGsXAuakWo6895el2iw61m+3q
+I/wBR+sY1tOEygnJ/eCFJbNkk7FrpzoOf5HwKRYIWGnkliA6OGE5xqBa4MRQKHLXA5xCieQeW6YInn5hiQDlcW45
odgkdY8CEUYo75zSXioCiSBg1OMarts0UK71cqLUOlm+tZrBge38NlZpuGP4mAXQw8V4HzxvBiwwYE3MrTVwxA2AiuJ
v8Y8AI14OjrEivnVSN9Y+FEVhKWmcZxOjICY3IW2acBeuTEGrYaOhB3zjaqb2bX6RiFN6KBoPA2rowuwilM+kY5ZfeL1O
ibs58uP6zSYtdacOckuxYOC+OcVdkHvYj8cYRK9juDL0m44f6LF2PkWtj8gJ9Whrb0RAIP93g5N9LFX2YFHXFdSjo8
YNdk+Vdxw3vBKW9acflhSADIIE614zUZHf/v1kuOYXUNtv6oIFk5qj6lgbLfcFkQ2UuXzjKk7rUE7um7CYIbqG1qj
jB0QeR9AbGa+fXE426dsr/cJaCTX5zY2z/sYqHGujsrN5LoWIEI4aGesKksGaATACB1nz9VZda/rZkppZwvndnupxh
Eb7J1IS53Aep4qHjOSBOX+8q/QP8wUqUCAIcJzb3lXDsU9G0Mb5msGeIha6epDQVzDC0r9tCbvgz/Z" />
</body>
</html>
```

Como ven, en ambos casos las imágenes se definen dentro del mismo archivo en lugar de apuntar a archivos externos, como hacemos normalmente.

*Para terminar este tedioso artículo, les muestro el código php con que codifiqué las imágenes en base64:

```
<?php
echo base64_encode(file_get_contents('img01.jpg'));
?>
```

Artículo por **Andrés Fernández**

Segmentación por país para OpenAds

Openads es un software creado en PHP que funciona como servidor de banners. Hemos

hablado repetidas veces de OpenAds DesarrolloWeb.com, un programa que antes se conocía con el nombre de PhpAdsNew.

- [Instalación de PhpAdsNew](#) (ahora OpenAds)

Ahora vamos a mostrar como poder configurar OpenAds para hacer segmentación por país, de modo que podamos definir el país o países a los que se distribuirán los banners.

De momento, hay que saber segmentar banners en OpenAds, es decir, aprender a definir las limitaciones de entrega de los banners. Esto lo vimos en el artículo [Segmentación de una campaña de banners](#).

Una vez que ya sabemos segmentar una campaña, a través de las limitaciones de entrega de los banners, ahora vamos a aprender a segmentar por país del visitante.

Para ello tenemos que hablar primero de las bases de datos de "Geotargeting", que son bases de datos que hacen corresponder un país a cada usuario a partir de su dirección IP. Existen diversas base de datos para averiguar el país de un usuario dada su IP, pero Openads trabaja con dos distintas:

- MaxMind <http://www.maxmind.com/>
- IP2Country <http://www.ip2country.com>

Nosotros hemos probado a utilizar la base de datos MaxMind, que tiene una versión gratuita que da bastante precisión al detectar el país.

Lo primero que tendremos que hacer es entrar en MaxMind y acceder a sus productos "Geolocation Technology". Veremos que hay un producto que se llama [Geolite Country](#). Este es gratuito, pero existen otras versiones de la base datos que tienen más precisión y ofrecen más datos sobre el usuario, incluso hay un producto que distingue incluso la ciudad del usuario.

De momento vamos a quedarnos con el producto gratuito (Open Source / Free), que servirá para nuestras pruebas. En adelante cada uno decidirá si necesita pagar por una base datos más precisa, que por suerte tiene un precio bastante razonable.

Nota: si queremos utilizar esta herramienta de Maxmind para la detección del país del usuario en cualquier aplicación PHP lee el artículo: [Detectar país del visitante](#)

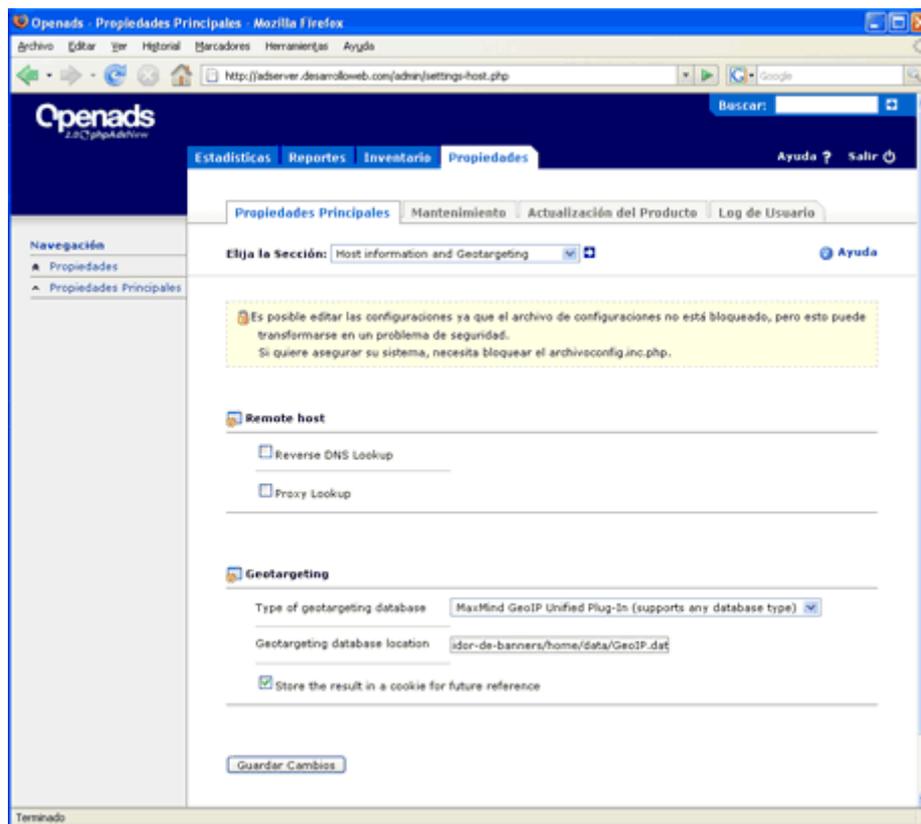
La base de datos que descargamos es la que pone "Download the latest GeoLite Country Binary Format", que nos permite descargar un archivo llamado GeoIP.dat.gz. Es un archivo comprimido que podemos descomprimir en nuestro ordenador antes de subirlo al servidor. (Podemos descomprimir archivos .gz con un programa como [7zip](#))

Una vez descomprimido lo subimos por FTP al directorio del servidor donde nos venga bien.

Configurar Openads para segmentar por IP del visitante

Con el archivo de la base de datos de Geolite Country (GeoIP.dat) cargado en el servidor, viene el paso en el que tenemos que configurar Openads. Para ello tenemos que entrar en openads e ir a la sección "Propiedades - Propiedades Principales".

En esa página tenemos un campo select desplegable, donde tenemos que seleccionar la opción que pone "Host information and Geotargeting". Entonces accederemos al formulario de configuración de la base de datos para el Geotargeting.



En el formulario tenemos al principio dos campos checkbox, para definir el comportamiento de Openads para obtener la IP del visitante.

Reverse DNS Lookup: Por si queremos que openads haga una búsqueda de DNS reverso para obtener datos del usuario.

Proxy Lookup: Para asegurarnos que la IP que detecta el servidor no corresponde a la IP de un Proxy, en caso que el visitante navegue utilizando un Proxy.

Las dos opciones se encuentran desactivadas por defecto. Esto es porque OpenAds tardaría bastante en hacer estas operaciones y se ralentizaría sensiblemente la distribución de banners. En principio los podemos dejar desactivadas y si tenemos problemas tal vez podemos probar a activarlas.

Luego está el campo para definir la base de datos de geotargeting que vamos a utilizar. El primer campo es un desplegable que pone "Type of geotargeting database". En ese desplegable hemos seleccionado la opción "MaxMind GeoIP Unified Plug-In (supports any database type)".

En el segundo campo, etiquetado con "Geotargeting database location" debemos escribir la ruta absoluta en nuestro servidor donde se encuentra el archivo GeoIP.dat, que habíamos subido por FTP. Será algo como esto:

```
/home/wwwroot/servidor-de-banners/home/data/GeoIP.dat
```

O como esto:

C:\Archivos de programa\EasyPHP1-8\www\openads\GeoIP.dat

Nota: Para saber la ruta absoluta de un directorio en PHP podemos consultar la variable `$_SERVER["DOCUMENT_ROOT"]`, que nos dará el directorio físico de la raíz de publicación de nuestro dominio. A partir de esa ruta podremos construir cualquier otra ruta que necesitemos dentro del sistema de directorios del servidor.

Luego damos al botón para guardar los cambios y si todo funciona bien ya tendremos configurado el geotargeting en openads.

Segmentar por país la distribución de un banner con Openads

Ahora veamos como definir que un banner sólo lo vean los usuarios de un país, por ejemplo, España.

Tenemos que ir a la pantalla de opciones de entrega del banner que deseemos segmentar y allí hay un select que tiene las distintas posibilidades para limitar la entrega.



Tenemos que escoger la que pone "Country", que sirve para limitar los países a los que se mostrará el banner.



Entonces podemos seleccionar los países a los que queremos limitar la entrega del banner, es decir, los países en los que se verá el banner.

Sólo nos queda probar si funciona la segmentación configurada.

Artículo por **Miguel Angel Alvarez**

Carro de Compras en PHP

Pueden ver el ejemplo en marcha de lo que vamos a hacer aquí.

Antes de empezar veremos, de manera resumida, para qué se utilizan las sesiones:

Las sesiones nos permiten registrar un número arbitrario de variables que se conservan durante toda la visita de un usuario a una página web. Dichas variables pueden ser diferentes para cada usuario, ya que están referenciadas por un identificador único que se le asigna a cada visitante. En otras palabras, una sesión es una manera de almacenar variables de manera temporal, semejante a una cookie, pero con ciertas diferencias: las cookies se almacenan en la

PC del usuario y pueden desactivarse; las sesiones, en cambio, se almacenan temporalmente en el servidor, en un fichero que se crea en el momento en que almacenamos la variable. Sabiendo esto, ya estamos en condiciones de entender cómo va a funcionar nuestro carro de compras.

Primero crearemos en mysql una tabla para nuestro catálogo de artículos:

```
create table catalogo (id int not null auto_increment primary key,producto varchar(100),precio decimal(9,2))
```

También vamos a necesitar algunas imágenes para ayudar visualmente al usuario que va a utilizar nuestro carrito (en el ejemplo vamos a utilizar estas, luego cada uno las reemplaza por las que quiera):

Imagen de producto no agregado al carrito:

productonoagregado.gif

Imagen de producto agregado:

productoagregado.gif

Botón para eliminar un producto del carrito:

trash.gif 

Botón para actualizar las cantidades de un producto agregado:

actualizar.gif 

Botón para continuar la selección de artículos:

continuar.gif 

Botón para ver el contenido del carrito:

vercarrito.gif 

Con estos elementos, podemos crear el archivo **agregacar.php**, que nos servirá para introducir productos dentro del carro (y para modificar sus cantidades) y que contendrá el siguiente código:

```
<?php
session_start();
//con session_start() creamos la sesión
//si no existe o la retomamos si ya ha
//sido creada
extract($_REQUEST);
//la función extract toma las claves
//de una matriz asociativa y las
//convierte en nombres de variable,
//asignándoles a esas variables
//valores iguales a los que tenía
//asociados en la matriz. Es decir,
//convierte a $_GET['id'] en $id,
//sin que tengamos que tomarnos el
//trabajo de escribir
//$id=$_GET['id'];
mysql_connect("localhost","usuario","password");
mysql_select_db("db");
//incluimos la conexión a nuestra
//base de datos
if(!isset($cantidad)){ $cantidad=1;}
```

```
//Como también vamos a usar este
//archivo para actualizar las
//cantidades, hacemos que cuando
//la misma no esté indicada sea
//igual a 1
$qry=mysql_query("select * from catalogo where
id='".$id."'");
$row=mysql_fetch_array($qry);
//Si ya hemos introducido algún
//producto en el carro lo
//tendremos guardado temporalmente
//en el array superglobal
//$_SESSION['carro'], de manera
//que rescatamos los valores de
//dicho array y se los asignamos
//a la variable $carro, previa
//comprobación con isset de que
//$_SESSION['carro'] ya haya sido
//definida
if(isset($_SESSION['carro']))
$carro=$_SESSION['carro'];
//Ahora introducimos el nuevo
//producto en la matriz $carro,
//utilizando como índice el id
//del producto en cuestión,
//encriptado con md5.
//Utilizamos md5 porque genera
//un valor alfanumérico que luego,
//cuando busquemos un producto
//en particular dentro de la
//matriz, no podrá ser confundido
//con la posición que ocupa dentro
//de dicha matriz, como podría
//ocurrir si fuera sólo numérico.
//Cabe aclarar que si el producto
//ya había sido agregado antes,
//los nuevos valores que le
//asignemos reemplazarán a los
//viejos.
//Al mismo tiempo, y no porque
//sea estrictamente necesario
//sino a modo de ejemplo,
//guardamos más de un valor en
//la variable $carro, valiéndonos
//de nuevo de la herramienta array.
$carro[md5($id)]=array('identificador'=>md5($id),
'cantidad'=>$cantidad,'producto'=>$row['producto'],
'precio'=>$row['precio'],'id'=>$id);
//Ahora dentro de la sesión
//($_SESSION['carro']) tenemos
//sólo los valores que teníamos
//(si es que teníamos alguno)
//antes de ingresar a esta página
//y en la variable $carro tenemos
//esos mismos valores más el que
//acabamos de sumar. De manera que
//tenemos que actualizar (reemplazar)
//la variable de sesión por la
//variable $carro.
$_SESSION['carro']=$carro;
//Y volvemos a nuestro catálogo de
//artículos. La cadena SID representa
//al identificador de la sesión, que,
//dependiendo de la configuración del
//servidor y de si el usuario tiene
```

```
//o no activadas las cookies puede
//no ser necesario pasarla por la url.
//Pero para que nuestro carro funcione,
//independientemente de esos factores,
//conviene escribirla siempre.
header("Location:catalogo.php?".SID);
?>
```

Luego creamos el archivo **borracar.php**, que nos permitirá eliminar artículos que hayamos ingresado:

```
<?php
session_start();
//con session_start()
//creamos la sesión si
//no existe o la retomamos
//si ya ha sido creada
extract($_GET);
//Como antes, usamos
//extract() por comodidad,
//pero podemos no hacerlo
//tranquilamente
$carro=$_SESSION['carro'];
//Asignamos a la variable
//$carro los valores
//guardados en la sesión
unset($carro[md5($id)]);
//la función unset borra
//el elemento de un array
//que le pasemos por
//parámetro. En este caso
//la usamos para borrar el
//elemento cuyo id le pasemos
//a la página por la url
$_SESSION['carro']=$carro;
//Finalmente, actualizamos
//la sesión, como hicimos
//cuando agregamos un producto
//y volvemos al catálogo
header("Location:catalogo.php?".SID);
?>
```

Luego creamos nuestro catálogo de productos, el archivo **catalogo.php**:

```
<?php
ob_start("ob_gzhandler");
//Las funciones ob_start y
//ob_end_flush te permiten
//escojer en qué momento
//enviar el resultado de un
//script al navegador. Si
//no las utilizamos estamos
//obligados a que nuestra
//primera línea de código
//sea session_start() u
//obtendremos un error
session_start();
//conectamos a la base de
//datos
mysql_connect("localhost","usuario","password");
mysql_select_db("db");
//rescatamos los valores
//guardados en la variable de
//sesión (si es que hay alguno,
```

```
// cosa que comprobamos con isset)
//y los asignamos a $carro.
//Si no existen valores, ponemos a false
//el valor de $carro
if(isset($_SESSION['carro']))
$carro=$_SESSION['carro'];else $carro=false;
//y hacemos la consulta
$qry=mysql_query("select * from catalogo order by producto asc");
?>
<html>
<head>
<title>CATÁLOGO</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<style type="text/css">
<!--
.catalogo {
font-family: Verdana, Arial, Helvetica, sans-serif;
font-size: 9px;
color: #333333;
}
-->
</style>
</head>
<body>
<table width="272" align="center" cellpadding="0" cellspacing="0" style="border: 1px solid #000000;">
<tr valign="middle" bordercolor="#FFFFFF" bgcolor="#DFDFDF" class="catalogo">
<td width="170"><strong>Producto</strong></td>
<td width="77"><strong>Precio</strong></td>
<td width="25" align="right"><a href="vercarrito.php?<?php echo SID ?>" title="Ver el contenido del carrito">
</a></td>
</tr>
<?php
//mostramos todos nuestros
//artículos, viendo si han
//sido agregados o no a nuestro
//carro de compra
while($row=mysql_fetch_assoc($qry)){
?>
<tr valign="middle" class="catalogo">
<td><?php echo $row['producto'] ?></td>
<td><?php echo $row['precio'] ?></td>
<td align="center">
<?php
if(!isset($carro[md5($row['id'])]['identificador')] || $carro[md5($row['id'])]['identificador']!
=md5($row['id'])){
//si el producto no ha sido
//agregado, mostramos la imagen
//de no agregado, linkeada
//a nuestra página de agregar
//producto y transmitiéndole a
//dicha página el id del artículo
//y el identificador de la sesión
?>
<a href="agregar.php?<?php echo SID ?>&id=<?php echo $row['id']; ?>">
</a><?php }else
//en caso contrario mostramos la
//otra imagen linkeada., a la
//página que sirve para borrar el
//artículo del carro.
{?><a href="borracar.php?<?php echo SID ?>&id=<?php echo $row['id']; ?>">
</a><?php } ?></td>
</tr><?php } ?>
</table>
</body>
</html>
```

```
<?php
ob_end_flush();
?>
```

Y por último el archivo en el que mostramos el contenido del carro, llamado **vercarrito.php**:

```
<?php
session_start();
//Iniciamos o retomamos la
//sesión
if(isset($_SESSION['carro']))
$carro=$_SESSION['carro'];else $carro=false;
//La asignamos a la variable
//$carro si existe o ponemos a false $carro
//en caso contrario
?>
<html>
<head>
<title>PRODUCTOS AGREGADOS AL CARRITO</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<style type="text/css">
<!--
.tit {
font-family: Verdana, Arial, Helvetica, sans-serif;
font-size: 9px;
color: #FFFFFF;
}
.prod {
font-family: Verdana, Arial, Helvetica, sans-serif;
font-size: 9px;
color: #333333;
}
h1 {
font-family: Verdana, Arial, Helvetica, sans-serif;
font-size: 20px;
color: #990000;
}
-->
</style>
</head>
<body>
<h1 align="center">Carrito</h1>
<?php
if($carro){
//si el carro no está vacío,
//mostramos los productos
?>
<table width="720" border="0" cellspacing="0" cellpadding="0" align="center">
<tr bgcolor="#333333" class="tit">
<td width="105">Producto</td>
<td width="207">Precio</td>
<td colspan="2" align="center">Cantidad de Unidades</td>
<td width="100" align="center">Borrar</td>
<td width="159" align="center">Actualizar</td>
</tr>
<?php
$color=array("#ffffff", "#F0F0F0");
$contador=0;
//las 2 líneas anteriores
//sirven sólo para hacer
//una tabla con colores
//alternos
$suma=0;
//antes de recorrer todos
//los valores de la matriz
```

```

//$carro, ponemos a cero la
//variable $suma, en la que
//iremos sumando los subtotales
//del costo de cada item por la
//cantidad de unidades que se
//especifiquen
foreach($carro as $k => $v){
//recorremos la matriz que tiene
//todos los valores del carro,
//calculamos el subtotal y el
// total
$subto=$v['cantidad']*$v['precio'];
$suma=$suma+$subto;
$contador++;
//este es el contador que usamos
//para los colores alternos
?>
<form name="a<?php echo $v['identificador'] ?>" method="post" action="agregacar.php?<?php echo SID ?>"
id="a<?php echo $v['identificador'] ?>">
<tr bgcolor="<?php echo $color[$contador%2]; ?>" class='prod'>
<td><?php echo $v['producto'] ?></td>
<td><?php echo $v['precio'] ?></td>
<td width="43" align="center"><?php echo $v['cantidad'] ?></td>
<td width="136" align="center">
<input name="cantidad" type="text" id="cantidad" value="<?php echo $v['cantidad'] ?>" size="8">
<input name="id" type="hidden" id="id" value="<?php echo $v['id'] ?>"> </td>
<td align="center"><a href="borracar.php?<?php echo SID ?>&id=<?php echo $v['id'] ?>"></a></td>
<td align="center">
<input name="imageField" type="image" src="actualizar.gif" width="20" height="20" border="0"></td>
</tr></form>
<?php
//por cada item creamos un
//formulario que submite a
//agregar producto y un link
//que permite eliminarlos
}
?>
</table>
<div align="center"><span class="prod">Total de Artículos: <?php echo count($carro);
//el total de items va a ser igual
//a la cantidad de elementos que
//tenga la matriz $carro, valor
//que obtenemos con la función
//count o con sizeof
?></span>
</div><br>
<div align="center"><span class="prod">Total: $<?php echo number_format($suma,2);
//mostramos el total de la variable
//$suma formateándola a 2 decimales
?></span>
</div><br>
<div align="center"><span class="prod">Continuar la selección de productos</span>
<a href="catalogo.php?<?php echo SID;?>">
</a>
</div>
<?php }else{ ?>
<p align="center"> <span class="prod">No hay productos seleccionados</span>
<a href="catalogo.php?<?php echo SID;?>">
</a>
<?php }?>
</p>
</body>
</html>

```

Y listo, ya tenemos nuestro carrito en funcionamiento.

Todos los archivos utilizados están disponibles para descargar gratuitamente desde el botón de descargas, ubicado en el lateral izquierdo de esta misma página.

*Artículo por **Andrés Fernández***

Conectar con Paypal

1) El envío de un mail al cliente con el detalle de su compra.

2) La integración con una Plataforma de Pagos.

1) El envío del mail es sencillo. La forma de armar la tabla de productos es muy similar a la que usamos para mostrarlos en el archivo `vercarrito.php`, de manera que no vamos a abundar en detalles.

La diferencia estriba en que guardamos todo el html que conformará el cuerpo del mensaje en una variable que llamamos `$html` y finalmente enviamos el mensaje con la función `mail`, la cual tiene como parámetros (en este orden) el e-mail del destinatario, el título del mensaje, el cuerpo del mensaje y las cabeceras que harán que el mensaje sea interpretado como HTML, además de proporcionar el nombre y la dirección de correo del remitente.

Archivo **mailer.php**

```
<?php
session_start();
$carro=$_SESSION['carro'];
//Guardamos en la
//variable $html el
//cuerpo del mensaje
$html = "
<html>
<head>
<title>Detalle de artículos Comprados</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<style type="text/css">
<!--
.tit {
font-family: Verdana, Arial, Helvetica, sans-serif;
font-size: 9px;
color: #FFFFFF;
}
.prod {
font-family: Verdana, Arial, Helvetica, sans-serif;
font-size: 9px;
color: #333333;
}
h1 {
font-family: Verdana, Arial, Helvetica, sans-serif;
font-size: 20px;
color: #990000;
}
-->
</style>
```

```
</head>
<body>
<table width=\"414\" border=\"0\" cellspacing=\"0\" cellpadding=\"0\" align=\"center\">
<tr bgcolor=\"#333333\" class=\"tit\">
<td width=\"198\">Producto</td>
<td width=\"107\">Precio</td>
<td width=\"109\" align=\"center\">Cantidad de Unidades</td>
</tr>";
$color=array("#ffffff","#F0F0F0");
$contador=0;
$suma=0;
foreach($carro as $k => $v){
$contador++;
$subto=$v['cantidad']*$v['precio'];
$suma=$suma+$subto;
$html.=<tr bgcolor=\"".$color[$contador%2]."\" class=\"prod\">
<td>".$v['producto'].</td>
<td>".$v['precio'].</td>
<td align=\"center\">".$v['cantidad'].</td>
</tr>";
}
$html .=
"</table>
<div align=\"center\"><span class=\"prod\">Total de Artículos: ".count($carro).</span> </div><br>
<div align=\"center\"><span class=\"prod\">Total: \$".number_format($suma,2).</span></div><br>
<div align=\"left\"><span class=\"prod\">
Aquí escribiremos un mensaje cualquiera, por ejemplo, cuáles son nuestras opciones de pago
</span></div>";
//Como queremos enviar el
//mensaje en formato html,
//colocamos las 2 cabeceras
//que nos permitirán hacerlo
$headers = "MIME-Version: 1.0\r\n";
$headers .= "Content-type: text/html; charset= iso-8859-1\r\n";
//Las siguientes 2 cabeceras,
//permitirán que el destinatario
//sepa a quién responder y
//quién le ha enviado el
//mensaje
$headers .= "Reply-To: mailremitente@dominio.com\r\n";
$headers .= "From: Nombre del Remitente<mailremitente@dominio.com>\r\n";
//En este ejemplo suponemos
//que el mail del destinatario
//lo hemos enviado desde un
//formulario con el método post,
//pero es indistinto desde donde
//se lo obtenga (consulta a la
//base de datos, almacenado en
//una variable de sesión,
//enviado por get,etc.)
mail("{$_POST['email']}", "Detalle de su compra en nuestro website", $html, $headers);
?>
```

2) Ya tenemos nuestro carrito y sabemos cómo enviarle un mail con los productos seleccionados al comprador, pero aún no hemos resuelto cómo haremos que éste nos pague.

Podríamos agregar, en el mail que hicimos anteriormente, algunas instrucciones para formalizar el pago a través de una transferencia o depósito bancario... es una opción.

Pero también podemos hacer que el cliente procese su compra antes de abandonar nuestra página.

Una manera de hacer esto último es integrar nuestro carrito con una Plataforma de Pagos

como Paypal. Para ello debemos sacar previamente una cuenta en <https://www.paypal.com/>, donde nos pedirán, entre otros requisitos, una dirección de correo electrónico que nos servirá para que el sistema de Paypal nos identifique como vendedores cuando un comprador realice una compra desde nuestro website.

Debemos tener en cuenta que, en el ejemplo que presentaremos, mientras se realice el proceso de pago el comprador oscilará entre nuestro servidor y el servidor de Paypal. Independientemente del protocolo que use nuestro servidor para las transacciones web (lo más probable es que use protocolo HTTP), Paypal usa protocolo HTTPS, que es más apropiado para el tráfico de información sensible, como por ejemplo números de tarjetas de crédito.

El esquema de trabajo que utilizaremos es el siguiente: en nuestro website procesaremos la selección de los artículos y calcularemos los precios. Luego derivaremos esa información y a nuestro cliente a Paypal. Ya en Paypal y bajo el paraguas del protocolo HTTPS, el cliente se identificará, verá los productos que ha seleccionado en nuestro website junto con el importe de los mismos, y elegirá alguna alternativa de pago o suspenderá la compra o el sistema le indicará que no tiene fondos suficientes o una larga lista de etcéteras. En cualquiera de todos estos casos, al terminar, Paypal redireccionará nuevamente al comprador a nuestro website.

En resumen, en el esquema descrito pueden presentarse básicamente dos alternativas: que el pago se concrete o que el pago no se concrete.

Primera alternativa: Se concreta el pago.



Si el pago se realiza correctamente, Paypal derivará al cliente a una página de nuestro site que nosotros le indiquemos para los casos de éxito en el pago (la llamaremos **ipn_success.php**), devolviéndonos al mismo tiempo algunas variables que nos servirán para realizar nuestros procesos: por ejemplo, si vendimos un acceso a un servicio web por 3 meses, actualizar la base de datos dándole permisos al comprador por ese período o enviar un mail a alguien diciéndole que debe entregar x detalle de productos a x cliente, descontar del stock, etc.

Segunda alternativa: No se concreta el pago.

Si el pago no se concreta, Paypal devolverá al cliente a una página de nuestro site que nosotros le indiquemos para los casos de fracaso en el pago (la llamaremos **ipn_error.php**). En esta oportunidad no deberemos procesar nada: sólo deberemos notificar al cliente que por alguna razón el pago no pudo concretarse.

Bueno, terminadas las explicaciones generales, vamos a los hechos. A la página **vercarrito.php** le agregaremos un botón que llamaremos "Finalizar Compra" el cual, al ser presionado, derivará a la página en la que procesaremos el envío a Paypal, a la cual llamaremos **regpago.php**.

Este es nuestro botón "Finalizar Compra": FINALIZAR LA COMPRA

La página **vercarrito.php** quedará así (Note que enviaremos el Total del Importe del carrito utilizando el método get):

```

<?php
session_start();
$carro=$_SESSION['carro'];
?>
<html>
<head>
<title>PRODUCTOS AGREGADOS AL CARRITO</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<style type="text/css">
<!--
.tit {
font-family: Verdana, Arial, Helvetica, sans-serif;
font-size: 9px;
color: #FFFFFF;

```

```

}
.prod {
font-family: Verdana, Arial, Helvetica, sans-serif;
font-size: 9px;
color: #333333;
}
h1 {
font-family: Verdana, Arial, Helvetica, sans-serif;
font-size: 20px;
color: #990000;
}
-->
</style>
</head>
<body>
<h1 align="center">Carrito</h1>
<?php
if($carro){
?>
<table width="720" border="0" cellspacing="0" cellpadding="0" align="center">
<tr bgcolor="#333333" class="tit">
<td width="105">Producto</td>
<td width="207">Precio</td>
<td colspan="2" align="center">Cantidad de Unidades</td>
<td width="100" align="center">Borrar</td>
<td width="159" align="center">Actualizar</td>
</tr>
<?php
$color=array("#ffffff", "#F0F0F0");
$contador=0;
$suma=0;
foreach($carro as $k => $v){
$subto=$v['cantidad']*$v['precio'];
$suma=$suma+$subto;
$contador++;
?>
<form name="a<?php echo $v['identificador'] ?>" method="post" action="agregar.php?<?php echo SID ?>"
id="a<?php echo $v['identificador'] ?>">
<tr bgcolor="<?php echo $color[$contador%2]; ?>" class='prod'>
<td><?php echo $v['producto'] ?></td>
<td><?php echo $v['precio'] ?></td>
<td width="43" align="center"><?php echo $v['cantidad'] ?></td>
<td width="136" align="center">
<input name="cantidad" type="text" id="cantidad" value="<?php echo $v['cantidad'] ?>" size="8">
<input name="id" type="hidden" id="id" value="<?php echo $v['id'] ?>"> </td>
<td align="center"><a href="borrar.php?<?php echo SID ?>&id=<?php echo $v['id'] ?>"></a></td>
<td align="center">
<input name="imageField" type="image" src="actualizar.gif" width="20" height="20" border="0"></td>
</tr></form>
<?php }?>
</table>
<div align="center"><span class="prod">Total de Artículos: <?php echo count($carro); ?></span>
</div><br>
<div align="center"><span class="prod">Total: $<?php echo number_format($suma,2); ?></span></div>
<br>
<div align="center"><span class="prod">Continuar la selección de productos</span>
<a href="catalogo.php?<?php echo SID;?>"></a>
<a href="regpago.php?<?php echo SID;?>&costo=<?php echo $suma; ?>"></a>
</div>
<?php }else{ ?>
<p align="center"> <span class="prod">No hay productos seleccionados</span> <a href="catalogo.php?<?php echo
SID;?>"></a>

```

```
<?php }?>
</p>
</body>
</html>
```

En la página **regpago.php**, como dijimos, recogeremos los productos y el Total del Importe y los enviaremos a Paypal valiéndonos de un formulario que contendrá varios campos ocultos cuya función explicaremos más tarde. En definitiva, la página **regpago.php** tendrá la siguiente estructura:

```
<?php
session_start();
//Asignamos todos los
//valores guardados en
//la sesión a la variable
//$carro, como hicimos en
//las páginas anteriores
$carro=$_SESSION['carro'];
//$products es la variable
//que usaremos para mostrar
//los productos en esta página
//(separados por '+')
$products="";
//$products2 es la que usaremos
//para enviar a Paypal
//(separados por ',')
$products2="";
foreach($carro as $k => $v){
$unidad=$v['cantidad']>1?" unidades de":" unidad de";
$products.=$v['cantidad'].$unidad.$v['producto']."+";
$products2.=$v['cantidad'].$unidad.$v['producto'].", ";
}
//eliminamos el último '+':
$products=substr($products,0,(strlen($products)-1));
//eliminamos la última coma
//y el espacio que sigue a
//la misma:
$products2=substr($products2,0,(strlen($products2)-2));
?>
<html>
<head>
<title>Finalizar Compra</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<style type="text/css">
<!--
.tit {
font-family: Verdana, Arial, Helvetica, sans-serif;
font-size: 9px;
color: #FFFFFF;
}
.prod {
font-family: Verdana, Arial, Helvetica, sans-serif;
font-size: 9px;
color: #333333;
}
h1 {
font-family: Verdana, Arial, Helvetica, sans-serif;
font-size: 20px;
color: #990000;
}
-->
</style>
</head>
```

```

<body>
<!-- Creamos el formulario para enviar a Paypal -->
<form action="https://www.paypal.com/cgi-bin/webscr" name="f1" id="f1" method="post">
<fieldset>
<legend class="prod"><strong>Finalizar la Compra</strong> <a href="#" onclick="javascript:window.open('https://
www.paypal.com/cgi-bin/webscr?cmd=xpt/popup/OLCWhatIsPayPal-outside','olcwhatispaypal','toolbar=no,
location=no, directories=no, status=no, menubar=no, scrollbars=yes, resizable=no, width=400, height=350');"></a></legend>
<input type="hidden" name="shipping" value="0">
<input type="hidden" name="cbt" value="Presione aquí para volver a www.nuestrositio.com >>">
<input type="hidden" name="cmd" value="_xclick">
<input type="hidden" name="rm" value="2">
<input type="hidden" name="bn" value="nombre de la empresa vendedora">
<input type="hidden" name="business" value="maildelvendedor@dominio.com">
<input type="hidden" name="item_name" value="<?php echo $products2; ?>">
<input type="hidden" name="item_number" value="Nombre del comprador">
<input type="hidden" name="amount" value="<?php echo number_format($_GET['costo'],2) ?>">
<input type="hidden" name="custom" value="<?php echo $_GET['costo'] ?>">
<input type="hidden" name="currency_code" value="USD">
<input type="hidden" name="image_url" value="">
<input type="hidden" name="return" value="http://www.nuestrodominio.com/ipn_success.php">
<input type="hidden" name="cancel_return" value="http://www.nuestrodominio.com/ipn_error.php">
<input type="hidden" name="no_shipping" value="0">
<input type="hidden" name="no_note" value="0">
<!-- Mostramos el detalle de la compra -->
<table width="50%" border="0" align="center" cellpadding="3" cellspacing="0" bgcolor="#EABB5D" style=" border-
color:#000000; border-style:solid;border-width:1px;">
<tr>
<td align="left" valign="top"><span class="prod"><strong>Detalle de los Productos
Seleccionados</strong> :</span><br>
<span class="texto1negro"> </span><span class="prod"><strong>Productos:</strong> <?php echo $products; ?
><br>
<strong>Precio Total:</strong> $<?php echo number_format($_GET['costo'],2) ?> </span></td>
</tr>
</table>
<input type="submit" name="Submit" value="Enviar">
</fieldset>
</form>

</body>
</html>
    
```

En **regpago.php** mostramos el detalle de los artículos y el total del importe que habíamos enviado desde **vercarrito.php**.

Pero veamos qué significan los campos hidden con los cuales trabajaremos (los que no mencionemos conviene dejarlos como están):

shipping: Es el costo de envío. Para nuestro ejemplo es cero, pero si no fuera ese el caso, aquí deberíamos colocar el importe correspondiente al total del costo de envío.

cbt: Es el mensaje que verá en Paypal el usuario al finalizar el proceso de pago.

rm: Es el método con que Paypal devolverá las variables a la página ipn_success.php (1 es get 2 es post).

bn: Esta es la identificación de la integración que estamos haciendo, normalmente la identificaremos con el nombre de la empresa vendedora.

business: Es el mail que el vendedor registró en su cuenta de Paypal.

item_name: Es el detalle de lo que estamos vendiendo.

item_number: Aquí va el número de item. Podemos completarlo con un código de operación o utilizarlo para otra cosa (personalmente lo utilizo para guardar el nombre del comprador, ya que cuando la operación se realiza y Paypal automáticamente genera sendos mails al vendedor y al comprador, suele incluir este dato y al vendedor le sirve para saber rápidamente quien es el comprador).

amount: Es el importe total de la operación.

custom: Aquí podemos colocar cualquier variable que luego necesitemos para realizar nuestros procesos cuando Paypal redirija al usuario a nuestra página de éxito.

currency_code: La moneda en que expresamos los valores:USD,GBP,JPY,CAD,EUR.

image_url: Es la ruta absoluta de la imagen que aparecerá en la cabecera de la página de Paypal cuando el comprador esté pagando. Se utiliza para que no perdamos del todo la identidad de nuestro site durante el proceso de pago, pero a menos que la imagen esté guardada en un servidor con protocolo HTTPS, es mejor dejar este campo en blanco ya que, si no lo hacemos de esa manera, cuando el comprador ingrese a Paypal le aparecerá un mensaje diciéndole que la página contiene elementos seguros y no seguros, cosa que puede asustar a algunos compradores.

return: Aquí colocaremos la ruta absoluta a la página ipn_success.php. Es la página a la que Paypal redirige al comprador si el pago se realiza correctamente, y a la que envía las variables que utilizaremos para los procesos ligados a la compra: generar un envío de productos, enviar un mail, descontar del stock, cambiar un nivel de acceso, etc.

cancel_return: Aquí colocaremos la ruta absoluta a la página ipn_error.php. Es la página a la que Paypal redirige al comprador si el pago no se realiza correctamente. En ella deberemos colocar un mensaje del tipo "Ocurrió un error y la operación no pudo realizarse..." para notificar el fallo al comprador.

Como vemos, Paypal actúa como un gran if:

```
if(pago==exitoso){enviar a ipn_success.php}
else{enviar a ipn_error.php}
}
```

De manera que no analizaremos demasiado la página **ipn_error.php**, ya que incluso puede ser estática. Veremos cómo podemos recuperar todas las variables que devuelve Paypal en **ipn_success.php**.

Si queremos saber todas las variables que Paypal nos envía, escribimos lo siguiente en el archivo **ipn_success.php** y luego realizamos una compra ficticia:

```
<?php
echo '<pre>';
print_r($_POST);
echo '</pre>';
?>
```

Si no queremos hacerlo, sabemos que la variable que siempre recibiremos es custom, y para recogerla podemos hacerlo así:

```
<?php
echo $_POST['custom'];
?>
```

En realidad, con custom es suficiente, ya que en ella podemos guardar todos los datos que necesitemos (es la ventaja de trabajar con el método post, con get estaríamos limitados). Por ejemplo, si hiciéramos lo siguiente:

```
<input type="hidden" name="custom" value="<?php echo $valor1."separator".$valor2."separator".$valor3; ?>">
```

Luego en ipn_success.php podríamos recuperar los valores de esta manera:

```
<?php
$vec=explode('separator',$_POST['custom']);
for($i=0;$i<count($vec);$i++){
echo '$valor1='.$vec[$i];
}
?>
```

Bueno, con esto tenemos un panorama general de cómo integrar nuestro carrito con Paypal. Sólo nos faltó señalar una cosa muy importante: Paypal proporciona una plataforma de pruebas en la que se pueden realizar operaciones ficticias, que funciona igual que la plataforma real, sólo que no hay dinero verdadero de por medio. Esto da la posibilidad de hacer todas las comprobaciones necesarias y solucionar todos los errores antes de poner online nuestra integración.

Dicha Plataforma de pruebas se llama Sandbox, y para poder utilizarla sólo hay que crearse una cuenta de pruebas en Paypal y cambiar el action de nuestro formulario en **regpago.php**, que era "https://www.paypal.com/cgi-bin/webscr", por este otro:

"https://www.sandbox.paypal.com/cgi-bin/webscr", y el mail de `<input type="hidden" name="business" value="maildelvendedor@dominio.com">` por el que consignemos en la cuenta de pruebas que creamos.

Para terminar, decir solamente que lo mejor es posibilitar más de una alternativa de pago. Podríamos, por ejemplo, hacer que el botón "Finalizar Compra" de **vercarrito.php** remitiese a una página en la cual se ofrecieran 2 alternativas de pago: una manual y otra automática. Si el usuario seleccionara la opción manual, le enviaríamos un mail con el detalle de los artículos como vimos en el punto **1)** y las instrucciones de pago o de cómo contactarse. Si, en cambio, optase por la manera automática, lo enviaríamos a la integración con Paypal que hemos visto en el punto **2)**.

*Artículo por **Andrés Fernández***

Como convertir páginas con extensión .PHP a la extensión que desees

Los requisitos para realizar este truco es tener un servidor Apache superior a la version 1.3. La solución es redefinir los procesos de ejecución asociados a los tipos de archivos mod_mime de Apache.

Después de esta breve introducción pasaremos a la explicación de este sencillo proceso de cambiar las extensiones .php por .wii en nuestro servidor.

Tenemos que indicar al Servidor que ejecute las páginas Web .wii como .php

La primera cosa que necesitaremos hacer es configurar Apache para permitir que utilicemos ".wii" como extensión.

Abrimos el archivo de Apache httpd.conf y vamos al área donde está AddType application/x-httpd-php .php y apenas agreguemos un .wii detrás de él. Debe parecer esto:

```
AddType application/x-httpd-php .php .ass
```

Una vez que hayas conseguido incorporar tu ".wii" en el archivo de httpd.conf, reinicia apache. Generalmente podría ser "reiniciar httpd" o "reiniciar apachectl", todo depende de tu sistema.

Ahora, una vez que cambiemos todas nuestras extensiones ".wii" de PHP en vez de ".php" podrás ver las páginas de esta manera:

"tusitio/index.wii" y funcionará justo como fuera "tusitio/index.php"

Otra cosa que podemos ajustar en el Apache en el archivo de httpd.conf es el ServerSignature a OFF y después corregir tu archivo de php.ini para incluir el "expose_php = off". Estos ajustes juntos harán más difícil para a los hackers descubrir la versión del servidor y otras informaciones valiosas que pueden poner en peligro tu servidor.

Resumen rápido de lo que acabamos de hacer:

1. Corregir httpd.conf para reflejar "AddType application/x-httpd-php .php .wii"
2. Corregir httpd.conf para reflejar "ServerSignature off"
3. Corregir el archivo de php.ini para reflejar el "expose_php = off"
4. Renombrar todos los archivos de .php a .wii y después fijar todos los enlaces para reflejar .wii en vez de .php
5. ¡Reiniciar Apache y prepararte para mostrar tu extensión .wii!

¡Buena suerte ocultando tu PHP!

Artículo por ***Manu Gutierrez***

CLASE class.ordenarCategorias.php

El ejemplo típico serán categorías de artículos en una tienda virtual de uso general, pero se puede adaptar a otro tipo de datos que deban seguir la misma pauta.

El Punto de partida

Supongamos que tenemos una tienda virtual, en la que vamos a vender un poco de todo: ordenadores, libros, discos, películas, etc. Lógicamente, contaremos con una tabla de

categorías en las que, posteriormente, encuadraremos los artículos de la tienda. La tabla de categorías contará, al menos, con los siguientes campos:

- Código de la categoría. A fin de que todo funcione adecuadamente, este será un campo numérico auto-incrementado.
- Nombre de la categoría.
- Código de la categoría padre. Este último es especialmente significativo en el planteamiento, ya que es el que va a permitir establecer la jerarquía en la que unas categorías dependerán de otras. Siguiendo con nuestro ejemplo, La categoría "Informática" podría contar, a su vez, con las categorías "Monitores", "Impresoras", etc. La categoría "Monitores" dependerá jerárquicamente de "Informática" y, a su vez, podrá contar con las categorías "Monitores CRT" y "Monitores TFT".

Si recuperamos los datos de la tabla de categorías, y los almacenamos en una matriz bidimensional (comportamiento típico de las consultas de selección SQL), podríamos obtener una matriz como la que se representa, con unas cuantas categorías de ejemplo, a continuación:

CODIGO DE LA CATEGORÍA A	NOMBRE DE LA CATEGORÍA A	CODIGO DE LA CATEGORÍA DE LA QUE DEPENDE
1	Informática	0
5	Libros	0
2	Monitores	1
8	Teclados	1
3	Monitores CRT	2
4	Monitores TFT	2
7	Históricos	5
6	Novela	5
9	Con cable	8
10	Inalámbricos	8
11	Inalámbricos con ratón	8

Las categorías pueden proceder de la tabla en cualquier orden arbitrario, según se hayan ido grabando. Cada categoría cuenta con dos códigos, tal como hemos mencionado anteriormente: el suyo propio y el de aquella de la que depende jerárquicamente. Como el código de cada categoría es un campo auto numérico, que se empiezan a crear desde 1, se ha reservado el código 0 para aquellas categorías raíces, que no dependen de ninguna otra.

El problema

Partiendo de que recuperemos los datos de la tabla en una matriz de memoria, es necesario

reorganizarlos en árbol. Por ejemplo, suponga que debemos mostrar las categorías en una lista de tipo SELECT, para que el administrador de la tienda elija en que categoría va a encuadrar un producto. La lista, tal como tenemos de momento la matriz, sería similar a la siguiente:

- Informática
- Libros
- Monitores
- Teclados
- Monitores CRT
- Monitores TFT
- Históricos
- Novela
- Con cable
- Inalámbricos
- Inalámbricos con ratón

Como ve, es poco práctica para el uso del administrador, y eso teniendo en cuenta las pocas categorías con las que estamos trabajando en nuestro ejemplo. Sería mucho más práctica si su aspecto fuera el siguiente:

- Informática
- Informática " Monitores
- Informática " Monitores " Monitores CRT
- Informática " Monitores " Monitores TFT
- Informática " Teclados
- Informática " Teclados " Con cable
- Informática " Teclados " Inalámbricos
- Informática " Teclados " Inalámbricos con ratón
- Libros
- Libros " Históricos
- Libros " Novela

Como ve, la lista es mucho más clara y coherente, ya que los nombres aparecen modificados y ordenados para que se vea el mapa de categorías en la lista, a primer golpe de vista. Sin embargo, es necesario que al cambiar el orden de los elementos, se mantenga la paridad entre cada categoría y los códigos asociados a la misma, como es lógico. Tenga en cuenta, que si construimos un SELECT con la matriz, los nombres serán lo que se mostrará al administrador, y los códigos de cada categoría serán los VALUES de cada OPTION.

Por lo tanto, la matriz debe poder modificarse, de modo que quede así:

```
array(11) {
  [0]=>
  array(3) {
    ["codigoCategoria"]=>
    string(1) "1"
    ["nombreCategoria"]=>
    string(11) "Informática"
    ["dependeDeCodigoCategoria"]=>
    string(1) "0"
  }
  [1]=>
  array(3) {
    ["codigoCategoria"]=>
    string(1) "2"
    ["nombreCategoria"]=>
```

```
string(23) "Informática " Monitores"
["dependeDeCodigoCategoria"]=>
string(1) "1"
}
[2]=>
array(3) {
["codigoCategoria"]=>
string(1) "3"
["nombreCategoria"]=>
string(39) "Informática " Monitores " Monitores CRT"
["dependeDeCodigoCategoria"]=>
string(1) "2"
}
[3]=>
array(3) {
["codigoCategoria"]=>
string(1) "4"
["nombreCategoria"]=>
string(39) "Informática " Monitores " Monitores TFT"
["dependeDeCodigoCategoria"]=>
string(1) "2"
}
[4]=>
array(3) {
["codigoCategoria"]=>
string(1) "8"
["nombreCategoria"]=>
string(22) "Informática " Teclados"
["dependeDeCodigoCategoria"]=>
string(1) "1"
}
[5]=>
array(3) {
["codigoCategoria"]=>
string(1) "9"
["nombreCategoria"]=>
string(34) "Informática " Teclados " Con cable"
["dependeDeCodigoCategoria"]=>
string(1) "8"
}
[6]=>
array(3) {
["codigoCategoria"]=>
string(2) "10"
["nombreCategoria"]=>
string(37) "Informática " Teclados " Inalámbricos"
["dependeDeCodigoCategoria"]=>
string(1) "8"
}
[7]=>
array(3) {
["codigoCategoria"]=>
string(2) "11"
["nombreCategoria"]=>
string(47) "Informática " Teclados " Inalámbricos con ratón"
["dependeDeCodigoCategoria"]=>
string(1) "8"
}
[8]=>
array(3) {
["codigoCategoria"]=>
string(1) "5"
["nombreCategoria"]=>
string(6) "Libros"
["dependeDeCodigoCategoria"]=>
```

```
string(1) "0"
}
[9]=>
array(3) {
["codigoCategoria"]=>
string(1) "7"
["nombreCategoria"]=>
string(19) "Libros " Históricos"
["dependeDeCodigoCategoria"]=>
string(1) "5"
}
[10]=>
array(3) {
["codigoCategoria"]=>
string(1) "6"
["nombreCategoria"]=>
string(15) "Libros " Novela"
["dependeDeCodigoCategoria"]=>
string(1) "5"
}
}
```

La solución

La clase `class.ordenarCategorias.php` nos proporciona una forma muy simple de obtener este resultado. Para usarla, debemos empezar, lógicamente, por incluirla en nuestro script (con PHP 5 podemos evitar la inclusión específica si lo deseamos, mediante `__autoload()`, pero eso es otra cuestión).

```
include ("class.ordenarCategorias.php");
```

A continuación, deberemos crear un objeto de la clase, así:

```
$matriz = new ordenarCategorias();
```

En este ejemplo, al objeto le hemos llamado `$matriz`.

Lo siguiente es cargar en la propiedad `matriz` del objeto la matriz original de la que partiremos, que está sin ordenar, tal como vemos a continuación:

```
$matriz->matriz = $matrizOriginal;
```

A continuación, debemos indicar, en la propiedad `idItem` cual es la clave asociativa de la matriz que contiene el código de cada categoría, así:

```
$matriz->idItem = "codigoCategoria";
```

El siguiente paso es cargar, en la propiedad `nombreItem`, el nombre de la clave asociativa que contiene el nombre de cada categoría, como vemos aquí:

```
$matriz->nombreItem = "nombreCategoria";
```

En la propiedad `idPadre` colocaremos la clave asociativa que usemos para identificar el código del que depende cada categoría, así:

```
$matriz->idPadre = "dependeDeCodigoCategoria";
```

Por último, usaremos el método `crearMapa()`, del objeto, que nos devuelve la matriz

modificada y ordenada según nuestros deseos:

```
$matrizOrdenada = $matriz->crearMapa();
```

Como puedes ver, el uso no puede ser más sencillo.

Como complemento, aclarar que si cada categoría en la matriz original tiene más datos asociados, estos no sufrirán ningún efecto colateral.

*Artículo por **Sunflower***

Estilos CSS distintos a una página con PHP y cookies

Presentamos un taller de PHP y cookies en el que vamos a ilustrar con un ejemplo el uso de Cookies en PHP. Es un ejemplo sencillo, porque PHP proporciona unas herramientas para el control de cookies muy simples de usar.

En este taller vamos a crear una página que puede configurarse con distintos estilos CSS. El usuario es quien decide qué aspecto desea que tenga la página, por medio de un formulario. Luego la página es capaz de recordar, entre los distintos accesos que realice el usuario, el aspecto que había elegido para mostrar la web.

Antes de empezar, vamos a recordar algunas cosas de las cookies en PHP. Las cookies son pequeñas informaciones de texto que se pueden almacenar en el navegador del visitante, siempre que este tenga configurado el browser para aceptarlas. Las cookies por tanto sólo pueden almacenar caracteres, pero aun así resultan de vital importancia para recordar estados o variables de una visita a otra del usuario.

Si queremos saber conceptualmente más sobre las cookies tenemos un artículo específico para aprender: [Qué son las cookies](#)

Tenemos otras informaciones sobre cookies en el [manual de PHP](#).

Si habemos leído los anteriores artículos o ya sabemos algo sobre cookies y PHP no tendremos problema en entender este taller. Podemos entonces pasar directamente a ver el ejemplo que vamos a crear en este ejercicio de PHP. Pero antes, dejamos el [enlace para que se pueda ver en funcionamiento](#) y así tener más claro cuál es el objetivo buscado.

El formulario HTML para seleccionar el estilo

```
<form action="taller-cookies-php.php" method="post">
Aquí puedes seleccionar el estilo que prefieres en la página:
<br>
<select name="estilo">
<option value="verde">Verde
<option value="rosa">Rosa
<option value="negro">Negro
</select>
<input type="submit" value="Actualizar el estilo">
</form>
```

Es un formulario simple, con un select para elegir entre los distintos estilos css disponibles para la página.

Recibir el formulario y crear la cookie

Si recibimos un valor del formulario tendríamos que crear una cookie para recordar qué estilo desea el usuario para visualizar la página.

```
//Veo si recibo datos del formulario
if(isset($_POST["estilo"])){
//es que estoy recibiendo un estilo nuevo, lo tengo que meter en las cookies
    $estilo = $_POST["estilo"];
    //meto el estilo en una cookie
    setcookie("estilo", $estilo, time() + (60 * 60 * 24 * 90));
}
```

En este ejemplo, la cookie la guardamos con el nombre "estilo" y con el valor captado del formulario. Asimismo hay que destacar que hemos configurado la cookie para que permanezca en el ordenador del usuario 90 días.

Ahora, si la página no ha recibido por post el estilo con el que se debe mostrar, tendremos que acceder al array \$_COOKIE para ver si está creada la correspondiente galletita y saber el estilo que el visitante había configurado en anteriores accesos.

```
}else{
//si no he recibido el estilo que desea el usuario en la página, miro si hay una cookie creada
if (isset($_COOKIE["estilo"])){
//es que tengo la cookie
    $estilo = $_COOKIE["estilo"];
}
}
```

Para acceder a la cookie lo hacemos con \$_COOKIE["estilo"].

En cualquier caso, el estilo lo hemos guardado en una variable global al script PHP \$estilo, que la vamos a utilizar para crear el enlace con la hoja de estilos CSS deseada.

```
if (isset($estilo)){
    echo '<link rel="STYLESHEET" type="text/css" href="" . $estilo . ".css">';
}
```

Si hay un estilo definido en \$estilo, pues hacemos la inclusión del mismo con la correspondiente etiqueta HTML.

Es muy sencillo el ejemplo, como se ha podido ver. Sólo queda mostrar el código PHP completo:

```
<?
//Veo si recibo datos del formulario
if(isset($_POST["estilo"])){
//es que estoy recibiendo un estilo nuevo, lo tengo que meter en las cookies
    $estilo = $_POST["estilo"];
    //meto el estilo en una cookie
    setcookie("estilo", $estilo, time() + (60 * 60 * 24 * 90));
}
else{
//si no he recibido el estilo que desea el usuario en la página, miro si hay una cookie creada
if (isset($_COOKIE["estilo"])){
//es que tengo la cookie
```

```
$estilo = $_COOKIE["estilo"];
}
}
?>
<html>
<head>
<title>Cookies en PHP</title>
<?
//miro si he tengo un estilo definido, porque entonces tengo que cargar la correspondiente hoja de estilos
if (isset($estilo)){
    echo '<link rel="STYLESHEET" type="text/css" href="' . $estilo . '.css">';
}
?>
</head>

<body>
Ejemplo de uso de cookies en PHP para almacenar la hoja de estilos css que queremos utilizar para definir el aspecto
de la página.
<p>
<form action="taller-cookies-php.php" method="post">
Aquí puedes seleccionar el estilo que prefieres en la página:
<br>
<select name="estilo">
<option value="verde">Verde
<option value="rosa">Rosa
<option value="negro">Negro
</select>
<input type="submit" value="Actualizar el estilo">
</form>

</body>
</html>
```

De nuevo, dejamos el link para [ver el ejemplo en marcha](#).

*Artículo por **Miguel Angel Alvarez***

Canonizar las URL de un dominio con PHP, elegir utilizar o no las www.

Recientemente se ha dado a conocer que motores de búsqueda como Google penalizan sitios web con contenido duplicado. Esto lo hemos hablado ya en el manual de Posicionamiento en buscadores, en el artículo [Sitios con contenido duplicado pueden ser penalizados en Google](#).

En ese artículo comentamos varios consejos para no ser penalizados por tener contenido duplicado en nuestro sitio.

Uno de los consejos es que nuestro sitio no se pueda acceder a través de URLs distintas, con o sin las 3 w. Por ejemplo:

<http://www.misitiodeejemplo.com>
<http://misitiodeejemplo.com>

Si ambas direcciones tienen el mismo contenido, Google puede pensar que estás duplicando información en tu dominio y puede caer la clasificación en buscadores de tu sitio web.

Entonces hemos creado un código en PHP que presentamos en este artículo para canonizar las direcciones URL de tu dominio (elegir cuál preferimos que sea utilizada e indexada en Google, la URL con las 3w o sin ellas), creando un dominio principal y redirigiendo las URLs de los dominios secundarios al principal. En este caso hemos preferido que el dominio principal sea el que tiene las 3w y que el acceso al dominio sin las www se redirija al dominio con ellas.

En realidad el código es muy simple. Sólo hay que conocer unas cuantas variables del array de variables superglobales `$_SERVER`, que contienen información sobre el servidor, la URL a la que se está accediendo, el dominio, etc.

Referencia: Para saber más de las variables de `$_SERVER` en PHP podemos acceder a estas informaciones:

- [Variables de sistema en PHP](#)
- [FAQ Variables \\$ _SERVER en PHP](#)

Veamos el código PHP comentado de este taller:

```
//trozo de código para que siempre se visite el dominio con las 3w
if (substr($_SERVER["SERVER_NAME"],0,4) != "www."){
    //si el dominio al que intentamos acceder está sin las 3w
    //lo redirijo al dominio con las 3w
    header("HTTP/1.1 301 Moved Permanently");
    header("Location: http://www." . $_SERVER["SERVER_NAME"] . $_SERVER["REQUEST_URI"]);
}
```

En nuestro código hacemos lo siguiente: Detectamos en el nombre del servidor al que se accede, o sea, el nombre del dominio de la URL que estamos accediendo. Si no comienza por "www." es que no se está accediendo al dominio como queremos, es decir, con las 3 w.

En ese caso, simplemente hacemos una redirección 301 (movido permanentemente) a la dirección con las www. Para saber el servidor al que estamos accediendo utilizamos la variable `$_SERVER["SERVER_NAME"]` y para saber la página concreta a la que tenemos que dirigir dentro del dominio utilizamos la variable `$_SERVER["REQUEST_URI"]`.

Este script PHP sirve para cualquier dominio donde lo pongamos y cualquier página de ese dominio, pues está parametrizado a través de `$_SERVER` tanto el nombre del dominio como el nombre de la página a la que redirigir al navegador.

Con esto conseguiremos no tener duplicado el contenido de la web en dos dominios distintos, con y sin las www, con lo que nuestro sitio web PHP estará un poco más optimizado para buscadores.

Artículo por [Miguel Angel Alvarez](#)

Kses. Validador de etiquetas HTML para PHP

Vamos a comentar un script realizado con PHP que consiste en un filtro de código HTML / XHTML, para validar que sólo tenga las etiquetas que deseamos, borrando todos los tags indeseables que pueda tener.

Hay veces que tenemos formularios, para entrada de datos por parte de los usuarios, en los que se pueden escribir etiquetas HTML, por ejemplo para formatear el texto. En estos casos es muy probable que deseemos permitir sólo un conjunto de etiquetas HTML, rechazando el resto. Para ello, tendríamos que analizar el código HTML escrito por el usuario para eliminar todas las etiquetas no permitidas.

No dudamos que crear un script PHP para analizar un código HTML y eliminar sólo ciertas etiquetas sería una tarea cuando menos laboriosa. Pero con Kses, el script que estamos comentando, podemos hacer esto de una manera extremadamente sencilla. Veremos en este artículo un ejemplo de funcionamiento de Kses, para limpiar el código HTML de etiquetas no permitidas.

Lo primero sería acceder a la página del proyecto en Source Forge:
<http://sourceforge.net/projects/kses>

Allí tendremos que descargar la versión más actualizada de Kses, en el momento de escribir el artículo iban por la release 0.2.2.

Kses tiene una función principal, que es la que realiza el filtro de etiquetas HTML. Esta función recibe tres parámetros:

- Código HTML / XHTML a validar. Es una cadena que contiene un texto mezclado con etiquetas, que es el que hay que filtrar para dejar sólo los tag permitidos.
- Etiquetas HTML permitidas. Es un array asociativo, de dos dimensiones, donde se especifican las etiquetas HTML válidas y los atributos que deseamos permitir.
- Protocolos permitidos, entre los que hay http, https, ftp, news, etc. Este campo es opcional y si no lo indicamos ya se cargan una serie de protocolos habituales. Estos protocolos se utilizan en los enlaces o las rutas de las imágenes, por ejemplo.

Pongamos que tenemos esta cadena de caracteres de código HTML:

```
$texto = "<a href='hola.php' target='_blank'>enlace</a>. Esto es un <b style='font-size:120%'>texto</b>. Esto <i>italico</i>.<p>Lo que sea<sup>3</sup></p>";
```

Ahora tenemos este array de etiquetas permitidas:

```
$tags_permitidos = array('b' => array(), 'i' => array());
```

Es este array de tags HTML válidos sólo estamos permitiendo etiquetas `` y etiquetas `<i>`, sin ningún atributo posible. Luego veremos la especificación de una lista mayor de etiquetas y atributos.

Para filtrar ese código HTML ejecutamos la función `kses()`:

```
$texto_filtrado = kses($texto,$tags_permitidos);
```

Esto nos devolvería una cadena como esta:

enlace. Esto es un `texto`. Esto `<i>italico</i>` .Lo que sea3

Vemos que sólo se han conservado las etiquetas que habíamos definido en el array y que a la etiqueta `` aparece sin el parámetro que tenía en el código HTML original.

Ahora veamos un array de etiquetas permitidas más amplio:

```
$tags_permitidos = array('b' => array('style' => 1),  
    'span' => array('style' => 1),  
    'i' => array('style' => 1),  
    'a' => array('style' => 1, 'href' => 1, 'title' => 1),  
    'p' => array('style' => 1, 'align' => 1),  
    'br' => array(),  
    'blockquote' => array('style' => 1),  
    'li' => array('style' => 1),  
    'ul' => array('style' => 1),  
    'ol' => array('style' => 1),  
    'u' => array('style' => 1),  
    'table' => array('cellpadding' => 1, 'cellspacing' => 1, 'border' => 1, 'style' => 1),  
    'tr' => array('style' => 1),  
    'td' => array('valign' => 1, 'align' => 1, 'rowspan' => 1, 'colspan' => 1, 'style' => 1)  
);
```

En este caso, estamos permitiendo unas cuantas etiquetas y atributos. Si filtramos el texto inicial con el código HTML de antes nos dará este resultado:

```
<a href='hola.php'>enlace</a>. Esto es un <b style='font-size:120%'>texto</b>. Esto  
<i>italico</i>.<p>Lo que sea3</p>
```

Como podemos ver, sólo ha eliminado la etiqueta <sup> y el atributo target='_blank' del enlace, que no estaban en el array de configuración de etiquetas HTML permitidas.

Espero que estas notas hayan sido suficientes para entender rápidamente kses, un útil filtro de etiquetas HTML, que podemos utilizar en nuestras aplicaciones PHP.

*Artículo por **Miguel Angel Alvarez***

Arreglar error con acentos en Google Coop

Google Coop es un buscador personalizado que ofrece Google como servicio, para incorporar en una página web un motor de búsqueda que muestra resultados en las webs que se indique. En un artículo anterior de DesarrolloWeb.com ya publicamos algunas informaciones sobre este interesante servicio: [Google Co-op](#), tu propio buscador personalizado.

En este artículo vamos a hablar del tema de los acentos en el sistema de Google Coop, que da errores poco descriptivos que pueden hacer creer al usuario que tu página no funciona correctamente. Podéis comprobarlo por vosotros mismos, si buscamos con una palabra acentuada en DesarrolloWeb.com, que utiliza el buscador de Google Co-op, nos devuelve una página vacía. No ofrece ningún error, pero tampoco ofrece ningún resultado, por lo que puede parecer que la página no está funcionando. (En nuestro caso mostramos un error por nuestra cuenta, pero google coop no muestra nada por el solo)

El problema de los acentos es debido al juego de caracteres que estamos utilizando. En realidad, el sistema de Google Coop funciona perfectamente con los acentos, lo que pasa es que tenemos que utilizar el juego de caracteres utf-8.

Si metemos esta línea en la cabecera del html, dentro de HEAD, podremos ver como el buscador, aunque busquemos palabras con acentos, funciona correctamente:

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

Con ello estamos marcando que el juego de caracteres debe ser utf-8.

El problema de utilizar utf-8 es que a menudo los acentos se ven de manera incorrecta, con un carácter raro. A no ser que utilicemos [caracteres especiales](#), los acentos que pongamos en la página se verán transformados en muchos casos a un código raro que queda muy feo. Esto también tiene que ver con el editor web que estemos utilizando y si tiene compatibilidad con el juego de caracteres UTF-8, así permitiría a los navegadores tratar bien los acentos al trabajar con UTF-8.

Está claro que todos deberíamos utilizar caracteres especiales para asegurarnos que los acentos y otros caracteres se ven correctamente en todos los casos, pero a veces, por la razón que sea, no los utilizamos. En DesarrolloWeb.com, de hecho, para no tener problemas con los acentos utilizamos el juego de caracteres ISO-8859-1. Nosotros no siempre utilizamos caracteres especiales y además la codificación de nuestros archivos de texto con código o con los manuales está en ANSI de 8 bit (un problema que arrastramos desde hace mucho tiempo y requeriría volver a codificar toda la página y revisar todos los artículos). El problema es que con el buscador Google Coop ISO-8859-1 no funciona bien y los acentos se le atragantan, y con ello se desata el problema que comentaba al principio.

Entonces ¿Qué hacer para que Google Coop funcione correctamente y que nuestra página también muestre bien los acentos? ¿Qué juego de caracteres utilizar UTF-8 o ISO-8859-1? Pues voy a contar la solución que he tomado yo, que quizás no sea la mejor y sea una simple chapuza, pero los resultados, más o menos, me parecen satisfactorios.

La en la página web en general utilizo el juego de caracteres ISO-8859-1, para lo que en la cabecera tenemos esta declaración:

```
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=ISO-8859-1">
```

Luego, cuando muestro las páginas del buscador CO-OP, utilizo el juego de caracteres UTF-8, con la declaración META vista al comienzo del artículo. Para saber si estoy o no en una página del buscador utilizo una variable global, aunque podría haber utilizado alguna de las variables de \$_SERVER. Hago algo como esto:

```
<?
if (isset($GLOBALS["estoy_en_buscador_google_coop"]))
    echo '<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />';
else
    echo '<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=ISO-8859-1">';
?>
```

En las páginas que quiero utilizar google coop declaro al comienzo una variable:

```
$estoy_en_buscador_google_coop=true;
```

De esta manera, una vez dentro del buscador ya utilizamos utf-8 y entonces cualquier búsqueda que se realice con o sin acentos dará resultados correctos.

El problema son las búsquedas realizadas desde cualquier otra de las páginas del sitio web, que tenían codificación ISO-8859-1. Entonces la cadena de búsqueda enviada al buscador con codificación ISO da problemas. Lo que hacemos, y aquí está la chapucilla, es buscar si la palabra que está buscando el usuario tiene acentos, porque entonces sabemos que Google

Coop no va a dar ningún resultado. Entonces en este caso lo que hacemos nosotros es mostrar un mensaje de error informando al usuario que tenemos un problema con los acentos.

```
function tiene_acentos($cadena){
    $acentos = "áéíóúüÁÉÍÓÚÛÑ";
    for ($i=0; $i<strlen($cadena); $i++){
        if (strpos($acentos, substr($cadena,$i,1))!==false)
            return true;
    }
    return false;
}

//echo "<!-- " . $_GET["q"] . "-->";

if (isset($_GET["dw"]) && tiene_acentos($_GET["q"])){
    echo "<p><br><br><i>Has intentado buscar una palabra con acentos</i></p>";
    echo "<p>Por favor, repite la búsqueda ahora para que funcione correctamente, que hemos activado la
compatibilidad con acentos. O bien, prueba a realizar la búsqueda de la palabra sin acentuar.</p><br><br>";
}
?>
```

Como las páginas del buscador ya las estamos mostrando con UTF-8, las nuevas búsquedas que realicen los usuarios con el buscador funcionarán correctamente. Así que en el mensaje que mostramos al usuario en caso de detectar un acento, nos permitimos el lujo de decir que hemos activado la compatibilidad con acentos y que pruebe a hacer la búsqueda de nuevo, con o sin acentos, que funcionará perfectamente.

Espero que hayamos podido aclarar nuestra solución y que no sea considerada demasiado chapucera. Aceptamos comentarios.

*Artículo por **Miguel Angel Alvarez***

Formato de números en PHP

Cuando mostramos un número en una página web podemos querer que tenga un formato específico. Por ejemplo, que tenga sólo dos decimales, o que utilice comas -o puntos- para separar decimales, así como las unidades de millar. Típicos formatos de número podrían ser:

```
1.000.505,56
5003.60
5,000.00
...
```

Presentar los números con uno u otro formato es sencillo, ya que en PHP existe una función específica para dar formato a los números, que dependiendo de los parámetros recibidos los formateará de una u otra manera. En este artículo vamos a conocer dicha función, así como sus posibilidades de configuración y ver varios ejemplos.

Función `number_format()`

El formateo de números lo llevaremos a cabo con la función de PHP `number_format()`. Esta función recibe uno, dos o cuatro parámetros. Es decir, tenemos estas restricciones:

- Hay un único parámetro obligado (el número que queremos formatear).
- El segundo parámetro es opcional, tal como el tercero y el cuarto
- Pero si especificamos el tercer parámetro, estamos obligados a especificar también el cuarto.

Veamos con detalle los parámetros de la función de formateo de números de PHP, con diversos ejemplos.

Parámetro 1, el número:

El primer parámetro es el número a formatear. Tal como dijimos, aunque resulta obvio, es un parámetro siempre necesario.

```
$numero = 15200.67;  
number_format($numero);  
//devuelve 15,201
```

En este caso, el formateo del número nos dará el número sin decimales y con una coma como separador de miles. Este formato es el que se utiliza en inglés (las personas de habla inglesa separan con comas los millares al escribir los números), que seguramente no nos sirva a los desarrolladores que trabajamos en español.

Cabe fijarse que la función `number_format()` ha realizado también un redondeo de los decimales que no está mostrando. Este redondeo lo vamos a ver bien en este ejemplo:

```
$numero = 999999999.99;  
number_format($numero);  
//devuelve 1,000,000,000
```

Parámetro 2, los decimales:

Con el segundo parámetro, que es opcional, indicamos el número de decimales que queremos que aparezcan en el número formateado.

```
$numero = 15200.67;  
number_format($numero,2);  
//devuelve 15,201.67
```

Como vemos, en este caso se han incorporado dos decimales al formato del número. Utiliza comas para separar los miles y un punto para separar las unidades de millar. Como vemos, sigue utilizando la notación inglesa para formatear números.

Otro ejemplo, en el que podemos apreciar que siempre se hace un redondeo del número, si los decimales a mostrar son menos que los que tiene el número original.

```
$numero = 1885200.89;  
number_format($numero,1);  
//devuelve 1,885,200.9
```

Parámetros 3 y 4, separadores de decimales y de unidades de millar

Los últimos parámetros, que debemos utilizar siempre juntos, sirven para especificar los separadores que queremos utilizar para los decimales y las unidades de millar. Si queremos formatear los números con la notación española tendríamos que utilizar forzosamente estos parámetros.

Por ejemplo, así haríamos para formatear los números en español:

- Separar los decimales con coma

- Separar las unidades de millar con un punto.

```
$numero = 1002002.365;  
number_format($numero, 2, ",", ".");  
//devuelve 1.002.002,37
```

Si, por ejemplo, no separar los millares, simplemente pasamos como separador de unidades de millar (cuarto parámetro) la cadena vacía:

```
$numero = 9540.2;  
number_format($numero, 2, "", "");  
//devuelve 9540,20
```

Conclusión sobre el formato de números con PHP

Eso es todo lo que tenemos que saber para formatear números en PHP. Como se ha podido ver es bien sencillo presentar los números con los decimales que necesitemos y con los separadores correctos de decimales y millares para el idioma español.

Artículo por [Miguel Angel Alvarez](#)

Leer un archivo traído por FTP con PHP

La gestión de archivos por PHP incluye la conexión por FTP o HTTP para abrir un archivo. Esto significa que se puede indicar una URL, en vez de la ruta del archivo en el sistema de archivos del servidor local. Con ello podremos abrir un archivo remoto, que se encuentra en otro servidor.

Para abrir un archivo debemos utilizar la función del sistema de archivos (Filesystem function) `fopen`. Esta función la describimos con detalle en el [manual sobre gestión de archivos por PHP](#). En este caso vamos a realizar un uso especial para conectar con ese archivo por FTP.

Antes que nada hay que decir que para que esto funcione, es decir, para poder conectar por FTP o por HTTP con un archivo remoto, tenemos que tener habilitada la directiva `allow_url_fopen` en el `PHP.ini` (`allow_url_fopen = on`).

La diferencia fundamental a la hora de conectar un archivo que se encuentra en un servidor remoto está en la línea del `fopen`:

```
$archivo = fopen ("ftp://usuario:clave@ftp.servidor.com/html/archivo.txt", "r");
```

Simplemente indicamos la ruta del archivo con una URL por el protocolo FTP. En dicha URL se indican los siguientes datos.

- `ftp://` es el protocolo.
- `usuario:clave` es el usuario y la clave del acceso FTP que estamos utilizando. Si fuera un ftp anónimo podríamos omitir estos datos.
- `@` para separar lo que es el usuario y la clave del nombre del servidor. También se debería omitir en el caso de un servidor anónimo.
- `ftp.servidor.com` es el nombre del servidor FTP al que estamos conectando
- `/html/archivo.txt` es la ruta desde el directorio raíz del FTP hacia el archivo que se desea

abrir.

Luego trataríamos el archivo de manera similar a como hemos visto en otros casos en el [manual sobre gestión de archivos por PHP](#).

Veamos un ejemplo completo de conexión por FTP para abrir, leer y mostrar un fichero de texto remoto:

```
<?php
$archivo = fopen ("ftp://user:password@ftp.server.com/html/probando.txt", "r");
if (!$archivo) {
echo "<p>No puedo abrir el archivo para lectura</p>";
exit;
}
$texto="";
while ($linea = fgets($archivo,1024)) {
if ($linea) $texto .= $linea;
}
echo $texto;
fclose ($archivo);
?>
```

Artículo por Miguel Angel Alvarez

Instalar PDT

Uno de nuestros lectores nos solicitó hace unas semanas una **guía para instalar PDT**, a raíz de nuestro artículo de presentación de [PDT, el IDE PHP basado en Eclipse](#).

A continuación publicamos la guía de instalación de este programa, que consta de una serie de pasos para descargar e instalar PDT.

En realidad es muy simple eso de instalar PDT... Sólo tienes que entrar en la página del [PDT Project](#) y hacer una pequeña búsqueda del archivo para instalar. De todos modos, te vamos a dar aquí unas guías para que lo puedas hacer sin problemas, o para que no te pierdas si no sabes inglés.

Una manera rápida de instalar PDT es instalar el paquete "All in One" de PDT. Este paquete te permitirá instalar todo lo que necesitas para comenzar a utilizar PDT, incluido el Eclipse y el propio módulo de desarrollo para PHP.

Lo que no instala el "All in One" es una versión de Java compatible con PDT o Eclipse. En el momento de publicar estas notas de instalación de PDT, la versión de Java que tenías que instalar es la Java 5 JRE o superior. Puedes conseguirla desde <http://java.com/es>

El siguiente paso es instalar ya el PDT "Todo en uno". Para ello entra en la página de descargas del PDT Project: <http://download.eclipse.org/tools/pdt/downloads/>

Entonces tienes que buscar la última release disponible que esté para descarga. Verás un recuadro que pone "Lastest Releases" y entonces seleccionas la última distribución que esté estable. "Stable Build". En el momento de escribir estas notas la última versión estable era la "1.0.3 Stable Build: S20080601-RC2".

Tienes que pulsar entonces esa distribución estable y entrarás a otra página donde aparece un epígrafe llamado "PDT All in One". Aquí aparecerá listado tu sistema operativo (Windows, Linux o Mac OS X), para que selecciones el archivo que necesitas dependiendo de tu sistema.

Luego ya apareces en la página de descargas, para que selecciones el mirror desde donde te van a enviar el archivo. Son poco más de 120 megas.

Una vez descargado, ya sólo te queda instalarlo. Para ello, nota que tu descarga, al menos en el caso de Windows, es un .zip. Dentro tienes un directorio llamado "Eclipse". Para instalar PDT tienes que descomprimir ese ZIP y colocar el directorio "eclipse" en cualquier lugar de tu disco duro. Puedes colocarlo en el directorio raíz de tu disco duro, pero si lo deseas dejar más ordenado, puedes meterlo en C:\Archivos de programa\

Entonces al descomprimir el ZIP se habrá creado, por ejemplo, la carpeta C:\Archivos de programa\eclipse y verás que dentro tienes un ejecutable llamado "eclipse.exe". Ese es el que tienes que abrir para poner en marcha el Eclipse con el PDT.

Si lo deseas, para poder ejecutar más cómodamente el PDT, puedes crear un acceso directo al ejecutable eclipse.exe y colocarlo dentro tu menú de inicio-programas, o sobre el escritorio o en el lugar o lugares que te venga más cómodo.

Para crear un acceso directo puedes hacerlo haciendo clic con el botón derecho del ratón encima del archivo y seleccionar la opción "Crear acceso directo". Luego puedes copiar y pegar ese acceso directo donde necesites, o arrastrarlo al lugar que desees.

Instalar PDT es fácil, lo que puede hacerse un poco más complicado es encontrar el lugar correcto para descargarlo. Esperamos haber dado una buena ayuda para conseguir instalar PDT de una manera sencilla.

*Artículo por **Miguel Angel Alvarez***

Instalar Zend Debugger para PHP

Uno de los pasos para proveer a nuestro servidor de PHP la posibilidad de hacer debug de aplicaciones creadas con PHP es instalar el Zend Debugger, una extensión para PHP que podemos obtener de manera gratuita. En este artículo veremos paso a paso la receta para conseguir instalar Zend Debugger.

El Zend Debugger lo vamos a utilizar en conjunto con el [IDE para desarrollo de aplicaciones en PHP PDT](#), que es una distribución de Eclipse preparada con todos los módulos para poder programar en PHP. Anteriormente ya publicamos un artículo sobre [cómo instalar PDT](#), que deberemos leer también en algún momento.

Primero, veamos el lugar donde conseguir el Zend Debugger: <http://downloads.zend.com/pdt/server-debugger/>

En esa página podremos ver varias distribuciones del programa para debug en el servidor de aplicaciones PHP. Tenemos que escoger la que sea adecuada para nuestro sistema operativo.

En mi caso, lo he instalado sobre Windows, en mi servidor para desarrollo local. He descargado entonces la primera opción ZendDebugger-5.2.14-cygwin_nt-i386.zip. Es un archivo comprimido que, si abrimos, veremos que tiene varias carpetas, con el debugger para varias versiones de PHP distintas. Tenemos que escoger la que nos corresponda conforme al servidor PHP que tengamos instalado. Por ejemplo, para mi PHP 5.2.6 tengo que utilizar la carpeta 5_2_x_comp.

Dentro hay un DLL, en el caso de Windows, que se tiene que copiar y pegar en el directorio donde están las extensiones de PHP del servidor. Este directorio lo podemos ver mediante un `phpinfo()`, que muestra la información completa sobre la instalación de nuestro PHP.

Para hacer este `phpinfo()`, si no lo sabes, tienes que crear un archivo en el directorio de publicación de páginas de tu servidor con el siguiente código:

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

Luego ejecutas ese archivo desde tu servidor local. Algo como `http://localhost/phpinfo.php`

Verás, entre la información que contiene, algo como esto:

error_reporting	6135	6135
expose_php	On	On
extension_dir	c:/wamp/bin/php/php5.2.6/ext/	c:/wamp/bin/php/php5.2.6/ext/
file_uploads	On	On
highlight.bg	#FFFFFF	#FFFFFF

Nota: Por facilitarme la vida, en mi ordenador local he instalado PHP por medio de Wamp Server, que instala de una vez todo lo necesario para trabajar con PHP. Tenemos descrito este proceso en nuestro artículo [Instalar PHP 5 fácilmente.](#)

Así pues, en el paso anterior hemos debido copiar el archivo `ZendDebugger.dll` (en el caso de Windows) en el directorio de extensiones de PHP que tengamos marcado al ver las características de nuestro PHP con `phpinfo()`. En mi caso he colocado el archivo dll en el directorio `c:/wamp/bin/php/php5.2.6/ext/`

El siguiente paso será colocar en el `php.ini` unas líneas para iniciar el Zend Debugger cuando se trabaje con PHP. Esas líneas de código las podemos ver en el archivo descargado del Zend Debugger, en el `README.TXT`.

El archivo `php.ini` de nuestro PHP también se puede localizar mediante el `phpinfo()`, donde aparece en la siguiente imagen:

Support	
Configuration File (php.ini) Path	C:\Windows
Loaded Configuration File	C:\wamp\bin\apache\apache2.2.8\bin\php.ini
PHP API	20041225
PHP Extension	20060613
Zend	20060510

De todos modos, para que se vea claro, las líneas que debes colocar en el php.ini, son las siguientes:

```
[Zend]
zend_extension_ts=[AQUÍ LA RUTA COMPLETA A LA DLL]
zend_debugger.allow_hosts=[AQUÍ LAS IPS DONDE DESEAMOS PERMITIR EL DEBUG]
zend_debugger.expose_remotely=always
```

Las líneas de configuración las podemos colocar en cualquier lugar del archivo php.ini, por ejemplo abajo del todo. Deberías personalizarlas para tu caso y quedaría con algo como esto:

```
[Zend]
zend_extension_ts=c:/wamp/bin/php/php5.2.6/ext/ZendDebugger.dll
zend_debugger.allow_hosts=127.0.0.1, 192.168.0.136
zend_debugger.expose_remotely=always
```

Como se puede ver, he colocado la ruta completa a la librería del Zend Debugger y luego una lista de Ips desde la que permitimos las conexiones para debug. La IP 127.0.0.1 es la dirección IP de localhost, que siempre es esa para todos los equipos. Es para decirle que permita debug desde el propio servidor donde tenemos PHP. Y luego, para asegurarme que funciona, he colocado 192.168.0.136 que es la IP de mi ordenador en mi red local, donde tengo el servidor PHP.

Luego tenemos que hacer, según el README.TXT, otro paso, que es colocar el archivo dummy.php, que viene en el fichero comprimido que hemos descargado antes, en la carpeta raíz de publicación del servidor web. Este paso yo me lo salté y aun así me funcionó, pero bueno.

En el PHP.ini nos recomiendan, según una documentación que he leído de PDT, alterar un par de configuraciones. Estas las comento por si acaso pudieran ser de utilidad o necesarias para nuestros casos, pero también me las he saltado y he podido hacer el debug PHP con PDT sin problemas.

```
implicit_flush = On
output_buffering = 0
```

Una vez hemos realizado estos pasos, debemos reiniciar el servidor PHP, es decir, reiniciar Apache o lo que tengamos como servidor, para que los cambios realizados en el php.ini se pongan en marcha.

Después de reiniciar el servidor debemos fijarnos que el Zend Debugger está funcionando, que podemos verlo en el phpinfo(). Debe aparecer un texto como este:

This program makes use of the Zend Scripting Language Engine:
Zend Engine v2.2.0, Copyright (c) 1998-2008 Zend Technologies
with Zend Debugger v5.2.12, Copyright (c) 1999-2007, by Zend Technologies

Powered By



Si no aparece ese texto, es que no está funcionando el Zend Debugger, por lo que algo hemos debido hacer mal en la instalación. Revisar los distintos pasos, fijándonos bien en el directorio donde hemos colocado la extensión de Zend Debugger (la ZendDebugger.dll en el caso de Windows) y la ruta de la extensión escrita en el php.ini.

Artículo por **Miguel Angel Alvarez**

Comprimir archivos Javascript desde PHP

Cuando se trata de optimizar una página web, cada Kilobyte cuenta. Como sabemos, cuanto menos pesada en bytes sea una página web, más rápido se transfiere por Internet y antes la puede visualizar el usuario. En este artículo vamos a ver una librería que nos puede ayudar a comprimir un archivo con código Javascript, para que ocupe menos espacio y se pueda transferir más rápidamente por la Red.

¿Cómo se puede comprimir un código Javascript?

El concepto que tenemos de compresión de archivos seguramente tenga relación con los conocidos .zip u otros formatos de compresión como el .rar. Pues la compresión que vamos a ver para códigos Javascript no tiene nada que ver. No os vamos a explicar que comprimáis el archivo con zip y lo publicuéis.

En realidad, en este caso, la compresión se basa en una optimización del código para que ocupe menos espacio. Como podemos saber, a la hora de programar, colocamos muchos caracteres de más en el código, que no son del todo necesarios. Esto ocurre con, por ejemplo, espacios de más, saltos de línea, tabulaciones, pero sobre todo con los comentarios al código.

JSMIn lo que hace es revisar el código Javascript, analizarlo y devolver una nueva versión de ese código, al que se le ha suprimido toda la información superflua. El resultado consiste en un código Javascript que sigue ejecutándose perfectamente en el navegador, con las mismas sentencias y funcionalidades, pero que ocupa mucho menos.

¿Dónde puedo obtener JSMin para PHP?

El script que estamos comentando, JSMin para PHP, podemos descargarlo gratuitamente desde una página de Google Code, donde está publicado para descarga:
<http://code.google.com/p/jsmin-php/>

Además, para el que le interese, dejamos el link a la página web oficial del producto, donde se ofrecen unas explicaciones básicas sobre qué hace JSMin para comprimir el código Javascript.

<http://www.crockford.com/javascript/jsmin.html>

¿Cómo comprimir el código con JSMIn para PHP?

JSMIn tiene una función, que debemos ejecutar para comprimir un código Javascript. Dicha función recibe un parámetro, que es una cadena con el código Javascript que queremos comprimir. Devuelve una cadena con el código, después de eliminar la información innecesaria y compactar las sentencias.

Lo primero de todo sería incluir la librería con el JSMIn para PHP, que hemos debido descargar desde la propia página web del producto, para obtener la versión más actualizada.

```
include ("jsmin-1.1.1.php");
```

Luego podemos utilizar la función minify de la clase JSMIn, que realiza el trabajo de comprimir el código JS.

```
$codigo_comprimido = JSMIn::minify("//codigo javascript");
```

Debemos pararnos para explicar esta línea de código, que puede sorprender por el operador :: de PHP (se ven los "::"?)

El operador :: de PHP sirve para invocar métodos de clases, lo que en algunos lenguajes se llaman métodos estáticos, con la particularidad que no hace falta tener ningún objeto instanciado de esa clase para poder ejecutar el método. Es decir, llamamos al método sin pasar por ningún objeto, simplemente con el nombre de la clase.

En ese caso, el include jsmin-1.1.1.php había definido una clase llamada JSMIn. Esa clase tiene un método llamado minify(). Con JSMIn::minify() estamos llamando a la función minify() declarada en la clase JSMIn, sin necesidad de haber creado ningún objeto de esa clase.

Script PHP para leer código Javascript de un archivo de texto y generar otro archivo de texto con el código comprimido

Para finalizar, voy a mostrar un código PHP que he creado para comprimir un archivo Javascript, basándome en JSMIn.

Este script define un nombre de archivo que se quiere comprimir y genera otro archivo en el mismo directorio que se llama igual, pero comenzando con "comprimido_". Ese archivo comprimido contiene el mismo código Javascript, una vez pasada la función minify() de JSMIn.

```
<?
$archivo_script_js = 'navegador.js';

include ("jsmin-1.1.1.php");
$codigo_comprimido = JSMIn::minify(file_get_contents($archivo_script_js));

$archivo = fopen("comprimido_" . $archivo_script_js, "w+");
fwrite($archivo, $codigo_comprimido);
fclose($archivo);
?>
```

Como se puede ver, se define arriba del todo el nombre del archivo que tiene el código PHP a comprimir.

Luego se realiza la compresión y se escribe los resultados en un archivo nuevo. El archivo antiguo se llama en este script "navegador.js" y el fichero nuevo con el script comprimido, que

se creará en el mismo directorio, se llama "comprimido_navegador.js".

Conclusión sobre comprimir los códigos JS

Ten en cuenta que la compresión no es un proceso reversible. Guarda siempre una copia de seguridad de los archivos con el código original, que la necesitarás si deseas editar el código.

Eso es todo. Espero que sea de utilidad esta clase para comprimir archivos Javascript desde PHP.

Artículo por Miguel Angel Alvarez

Convertir caracteres UTF-8 con PHP

En ocasiones los textos de la página web pueden contener caracteres raros en lugar de acentos u otras letras propias del idioma español, como la ñ o las letras con diéresis y de otros idiomas latinos. Esto suele ocurrir en casos en que la base de datos trabaje en un juego de caracteres distinto que la página, o que los datos estén mal cargados en la misma. Generalmente estos errores surgen por un tratamiento inadecuado del juego de caracteres UTF-8, que no están procesados como UTF-8, sino como si fueran ISO-8859-1. Con PHP podemos decodificar esos caracteres para que a la hora de mostrar los textos en la página aparezcan correctamente. Para ello existen una funciones que vamos a comentar en este artículo de DesarrolloWeb.com.

Cuando vemos incorrectamente un texto nos pueden aparecer textos como estos:

SoluciÃ³n Ãtil y apaÃ±ada a UTF-8

Este texto tiene caracteres codificados en UTF8, que debemos decodificar para que aparezcan correctamente los acentos y demás símbolos del alfabeto latino.

Decodificar una cadena UTF-8 con PHP - decode utf8 php

Existe una función de PHP que realiza el trabajo de conversión de los caracteres UTF8 a sus correspondientes códigos reales.

La función se llama `utf8_decode()` y lo que hace es convertir una cadena que realmente utiliza ISO-8859-1 pero que tiene caracteres codificados a con UTF-8. La conversión genera el string en ISO-8859-1 pero con un sólo byte para todos los caracteres, con lo que se verán correctamente.

Se utiliza así:

```
utf8_decode("SoluciÃ³n Ãtil y apaÃ±ada a UTF-8");
```

Devolverá una cadena el texto correctamente escrito:

Solución útil y apañada a UTF-8

Codificar una cadena al juego de caracteres UTF-8 - encode utf8 php

Existe una función para hacer justo el paso contrario, que vamos a ver también en desarrollo web .com. Es decir, partiendo de una cadena en el juego de caracteres ISO-8859-1, obtener la correspondiente traducción a UTF-8.

Esto lo podemos necesitar, por ejemplo, si nuestra base de datos está definida con UTF-8 y tenemos entrada de datos ISO-8859-1.

El uso es bien simple:

```
echo utf8_encode("Mañanas de programación PHP");
```

Y devolverá el correspondiente string convertido a UTF-8, que si lo mostramos en una página que utiliza el juego de caracteres ISO-8859-1, se vería de esta manera:

Mañanas de programación PHP

Artículo por [Miguel Angel Alvarez](#)

Convertir los caracteres especiales con PHP, sin alterar etiquetas HTML

Cuando hablamos de caracteres especiales del HTML, por si queda alguna duda, nos referimos a los caracteres que se tienen que sustituir en páginas web, dentro del código HTML, con sus correspondientes códigos del tipo `´` o `"`. Estos los debemos de conocer ya, pero para el que no sepa cuáles son les sugiero entrar en el artículo [Caracteres especiales](#).

Los caracteres especiales, en principio, deberían escribirse con sus correspondientes códigos, aunque muchas veces no se hace por comodidad o porque se ven correctamente en nuestro ordenador y pensamos que así de bien los vería cualquier persona. En este artículo de desarrollo web .com vamos a mostrar cómo convertir fácilmente los caracteres especiales del HTML por medio de una única función PHP, que recibirá el texto con acentos y caracteres para convertir y lo devolverá una vez procesado y convertidos los caracteres. La clave de este ejercicio es convertir dichos caracteres especiales, pero sin alterar las etiquetas HTML que pueda tener el texto. Luego explicaré esto con detalle.

Lo bueno de este script que vamos a comentar es que está creado haciendo uso de algunas funciones que trae PHP de casa, con lo cual, no tenemos por qué conocer la lista completa de los caracteres especiales. Simplemente utilizaremos algunas funciones interesantes del lenguaje.

Funciones de PHP para hacer traducción de caracteres especiales

En PHP existen varias funciones para cambiar caracteres por sus códigos especiales, pero ninguna de las funciones hace el trabajo justamente como deseaba. Primero veamos dichas funciones:

htmlentities(): esta función transforma todos los caracteres de un texto que tienen conversión a caracteres especiales. Esto incluye desde los acentos o la eñe a los mayores y menores que, las comillas, etc.

htmlspecialchars(): esta función convierte sólo los caracteres especiales que pueden afectar

a las etiquetas HTML, como mayor y menor qué, comillas simples y dobles y el carácter &.

La primera función, `htmlentities`, no me sirve porque mi texto contiene etiquetas HTML y no quería que se afectaran dichas etiquetas y que al poner el texto en la página se viera formateado con las etiquetas escritas. Si pasamos `htmlentities`, las etiquetas HTML luego serían interpretadas como texto. Por su parte `htmlspecialchars` tampoco me sirve, porque sólo convierte los caracteres que tienen que ver con las etiquetas, y son justo esos los que quiero dejar inalterados.

Así que vamos a ver una función nueva, no incluida en PHP pero que se basa en funciones del lenguaje, para convertir sólo los caracteres que nos interesan. Esta función la utilizamos en DesarrolloWeb.com y otros sitios de nuestros. Hay que decir que he tomado la inspiración para hacer esta función de un comentario que hay en la propia página de `php.net`.

Veamos el código PHP de la función:

```
function convertir_especiales_html($str){
    if (!isset($GLOBALS["carateres_latinos"])){
        $todas = get_html_translation_table(HTML_ENTITIES, ENT_NOQUOTES);
        $etiquetas = get_html_translation_table(HTML_SPECIALCHARS, ENT_NOQUOTES);
        $GLOBALS["carateres_latinos"] = array_diff($todas, $etiquetas);
    }
    $str = strtr($str, $GLOBALS["carateres_latinos"]);
    return $str;
}
```

Lo que hace la función, en líneas generales, es crear un array asociativo, con todos los caracteres que se desean convertir y su correspondiente carácter especial HTML. Luego se utiliza la función `strtr` para cambiar en un string todas los caracteres del array creado anteriormente.

Para obtener el array asociativo con todos los caracteres (en realidad sólo los que se traducen en códigos especiales) y sus correspondientes caracteres especiales, podríamos hacerlo a mano, definiendo por nosotros mismos la variable del array e insertando cada uno de los items. Pero en este ejemplo se utiliza otra función de PHP:

`get_html_translation_table()`: Esta función devuelve un array asociativo que se usa en las funciones `htmlspecialchars()` y `htmlentities()` para convertir esos caracteres. Recibe dos parámetros, aunque el segundo es opcional. El primer parámetro es una constante `HTML_ENTITIES` o `HTML_SPECIALCHARS`, para conseguir el array asociativo que utilizan cada una de esas funciones. El segundo parámetro sirve para indicar el estilo de comillas que se desea obtener en el array. Con la constante `ENT_NOQUOTES` indicamos que no ponga en el array las comillas simples ni las dobles.

Así pues, con las líneas:

```
$todas = get_html_translation_table(HTML_ENTITIES, ENT_NOQUOTES);
$etiquetas = get_html_translation_table(HTML_SPECIALCHARS, ENT_NOQUOTES);
```

Conseguimos dos arrays, uno con todos los caracteres HTML y otro con sólo los caracteres que afectan a las etiquetas. Luego, haciendo la diferencia de el primer array menos el segundo obtendremos el array con todos los caracteres que nos interesan.

Lo que veis de `$GLOBALS["carateres_latinos"]`, sirve para meter en una variable global ese array y luego no tener que volver a generarlo las sucesivas veces que se llama a la función.

Por último utilizamos la mencionada función `strtr()` para hacer la traducción de los caracteres contenidos en el array.

Una llamada a esta función como esta:

```
convertir_especiales_html('<p align="center">La ejecución de esta función sirve para <b>convertir caracteres
```

especiales con PHP sin tocar las etiquetas.
Parece que funcionó!</p>');

Devolvería un texto como el siguiente:

```
<p align="center">La ejecución de esta función sirve para <b>convertir caracteres especiales con PHP</b> sin tocar las etiquetas. <br>Parece que funciona!</p>
```

Quizás se me ha quedado un poco largo este artículo, aunque ya sabéis que en desarrollo web .com nos gusta explicar las cosas con calma para que se puedan entender perfectamente. Esperamos que esta función pueda resultar de utilidad.

*Artículo por **Miguel Angel Alvarez***

Implementar códigos BBCode en PHP con PEAR

Los códigos BBCode permiten a los usuarios crear un formato especial para textos que envían a foros o comentarios de post o páginas web. Son parecidos a etiquetas, con la particularidad que se escriben entre corchetes y permiten a los webmasters mantener un mayor control sobre el formato de las páginas, al ser posible restringir qué códigos BB se pueden utilizar, cuáles no y qué traducciones se deben utilizar para cada BBCode.

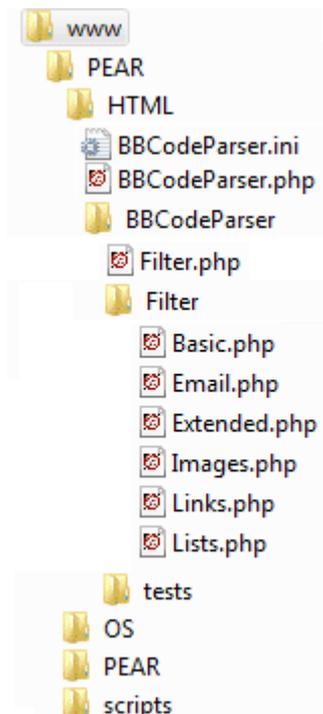
Si estás leyendo este texto posiblemente ya no necesites más explicaciones sobre lo que son los BBCode y quieras ver directamente cómo implementarlos en tus scripts PHP. No obstante, indicamos que en DesarrolloWeb.com hemos publicado una explicación detallada de los [BBCode](#) y tipos de formato que soportan.

En este artículo pretendemos explicar cómo implementar el soporte a BBCode en una página web, de una manera sencilla y muy depurada, a través de [PEAR](#), un framework PHP muy conocido y ampliamente utilizado.

PEAR dispone una clase que sirve para implementar los BBCode en PHP sin apenas esfuerzo, ya que provee todo lo que necesitamos para procesar textos que tengan esta notación y convertirlos a HTML entendible por navegadores. La clase que nos estamos refiriendo está en el paquete HTML_BBCodeParser y su documentación se puede acceder en la URL: http://pear.php.net/package/HTML_BBCodeParser

Nota: Habría que comentar que para utilizar PEAR es necesario tener instalado el framework en nuestro sistema e instalarlo no significa simplemente subir unos archivos por FTP al servidor, sino que hay que realizar algunas acciones que no vamos a explicar en este artículo. Sin embargo, puedes descargar por separado el paquete principal de PEAR y el paquete HTML_BBCodeParser y subirlos por FTP a tu servidor. Luego, con algunos pequeños cambios en los includes, para que se pueda acceder a los códigos según las rutas donde has colocado los directorios de los paquetes, podrías utilizar el paquete HTML_BBCodeParser sin problemas. Esto, hay que decirlo, es una gran chapuza, pero nosotros lo hemos hecho con éxito en un sitio web.

Para aclararnos, una vez instalado PEAR y el paquete HTML_BBCodeParser, tanto si lo hemos hecho con el instalador de PEAR como si lo hemos hecho "a mano", deberíamos tener una estructura de directorios como la siguiente:



Código PHP para implementar los BBCode

El primer paso sería incluir la librería o paquete para trabajar con BBCode de PEAR:

```
include ("PEAR/HTML/BBCodeParser.php");
```

El segundo paso sería instanciar un objeto de la clase que viene con el paquete HTML_BBCodeParser, que se llama con el mismo nombre del paquete:

```
$analizador_bbcode = new HTML_BBCodeParser();
```

Ahora tenemos un analizador o traductor de BBCode listo para funcionar.

Lo que necesitamos ahora es una variable con el texto a analizar y a traducir los BBCodes, por ejemplo podría ser lo siguiente:

```
$texto = "[b]negrita[/b] [i]cursiva[/i]";
```

Ahora tenemos que cargar ese texto y analizarlo, a través de distintos métodos del objeto de clase HTML_BBCodeParser, que habíamos instanciado antes:

```
$analizador_bbcode->setText($texto);  
$analizador_bbcode->parse();
```

Por último, podemos utilizar otro método del objeto de clase HTML_BBCodeParser para extraer el texto analizado, donde los BBCodes han sido traducidos por sus correspondientes etiquetas HTML.

```
$texto_traducido = $analizador_bbcode->getParsed();
```

El código PHP completo se puede ver a continuación:

```
include ("PEAR/HTML/BBCodeParser.php");
$analizador_bbcode = new HTML_BBCodeParser();
$texto = "[b]negrita[/b] [i]cursiva[/i]";
$analizador_bbcode->setText($texto);
$analizador_bbcode->parse();
$texto_traducido = $analizador_bbcode->getParsed();
```

En el [siguiente artículo de BB Codes con PEAR](#) que vamos a publicar en desarrolloweb.com veremos la manera de mejorar un poco este código y tratar de personalizar un poco los tipos de códigos BB soportados.

Sólo un único detalle antes de acabar. Se trata de explicar un método que puede simplificar un poco este código: método `qparse()`:

El código anterior podríamos reducirlo un poco con el método `qparse` (quick parse), que recibe directamente el texto a analizar y devuelve el texto con los BB Codes traducidos. Quedaría como sigue:

```
include ("PEAR/HTML/BBCodeParser.php");
$analizador_bbcode = new HTML_BBCodeParser();
$texto = "[b]negrita[/b] [i]cursiva[/i]";
$texto_traducido = $analizador_bbcode->qParse($texto);
```

*Artículo por **Miguel Angel Alvarez***

BBCode en PHP con PEAR, parte 2

En el anterior artículo estuvimos tratando sobre la [implementación de BB Codes en PHP con PEAR](#). Dejamos patente que, una vez instalado PEAR y el paquete HTML_BBCodeParser, es muy sencillo analizar un texto en busca de BB Codes y traducirlo a sus correspondientes etiquetas HTML. Pero aun nos queda algún detalle que sería bueno comentar.

Configurar los BB Codes soportados con el archivo BBCodeParser.ini

Existe un archivo de configuración que podemos cargar a la hora de instanciar el analizador de códigos BBCode, que sirve, entre otras cosas, para decir qué grupos de BBCode deseamos soportar.

Existen varios filtros para clasificar por grupos los BB Codes. Estos filtros están en las carpetas de instalación del paquete HTML_BBCodeParser, en el directorio Filter. Además, nosotros podríamos incluso crear nuevos filtros para crear nuevos códigos BBCode a traducir.

El caso es que, a través de un fichero de configuración, se puede decir cuáles de estos filtros se desean utilizar y cuáles no. Este fichero es BBCodeParser.ini, que debemos dejarlo en el directorio HTML que cuelga de PEAR, donde está también el archivo BBCodeParser.php.

Ese archivo .ini está en la descarga del paquete BBCodeParser de PEAR. dentro de la carpeta example. En el artículo anterior que publicamos en desarrollo web .com sobre los BB Codes con PEAR ya comentamos dónde descargar el paquete este y la estructura de directorios que

deberíamos tener en el servidor.

Si editamos ese archivo .ini veremos que tiene una línea donde se indican los ficheros que queremos soportar.

```
filters = Basic,Extended,Links,Images,Lists,Email
```

Con esa línea indicamos los filtros que se desean procesar, es decir, los grupos de BBCode que se tienen que analizar y cambiar por etiquetas HTML. (para saber qué etiquetas hay en cada tipo de BBCode, podemos leer el artículo [Qué es BBCode](#))

Nosotros simplemente indicaremos los filtros deseados, separados por coma.

Al instanciar el analizador de BBCode tenemos que indicarle que debe leer el BBCodeParser.ini, de la siguiente manera:

```
$opciones = parse_ini_file('PEAR/HTML/BBCodeParser.ini');  
$analizador_bbcode = new HTML_BBCodeParser($opciones);
```

Ahora podríamos generar una variable con un número mayor de BBCodes, ya que los soportamos todos, tal como se indicó en el archivo de configuración.

```
$texto = "[b]negrita[/b] [i]cursiva[/i] [url]http://www.desarrolloweb.com[/url] [u][b]Texto subrayado y negrita[/u]  
[/b] [code]esto es un codigo fuente[/code]";
```

Ahora veamos el código completo para analizarlo y mostrarlo en la página:

```
$texto = "[url]http://www.desarrolloweb.com[/url] [u][b]Texto subrayado y negrita[/u][[/b] [code]esto es un codigo  
fuente[/code]";  
include ("PEAR/HTML/BBCodeParser.php");  
$opciones = parse_ini_file('PEAR/HTML/BBCodeParser.ini');  
$analizador_bbcode = new HTML_BBCodeParser($opciones);  
$texto_traducido = $analizador_bbcode->qParse($texto);  
echo $texto_traducido;
```

Consejos para convertir los textos que vienen de formularios

Sólo queremos comentar que, a la hora de recibir datos de formularios, cargados por los usuarios, conviene antes hacer una conversión a caracteres especiales de ciertos caracteres, como mayores y menores que, con la función `htmlspecialchars()`. Como debemos saber si hemos seguido los manuales de PHP publicados en desarrolloweb.com, `htmlspecialchars()` eliminará el significado de cualquier etiqueta HTML que haya introducido el usuario, que serán tratadas como texto normal.

```
$texto_formateado = $parser->qParse(htmlspecialchars($texto));
```

Luego, a la hora de mostrar los textos formateados con BBCode, sólo quedaría convertir los saltos de línea en etiquetas BR, con la función de PHP `nl2br()`, como sigue:

```
echo nl2br($texto_formateado);
```

Otras implementaciones de BBCode en PHP

Hemos comentado en este artículo y el anterior, la implementación más rápida y más segura para procesar los BBCode. A través de PEAR tenemos la seguridad que el análisis de los

BBCodes será correcto y que el código resultante es completo y sin errores como etiquetas no cerradas.

Porque hay que decir que el paquete HTML_BBCodeParser está preparado para devolver sólo código XHTML validado correctamente. Si por ejemplo, al crear un texto con BBCodes, un usuario se olvida de cerrar una etiqueta, el sistema analizador la cerrará automáticamente.

Entendemos, no obstante, que puede ser difícil poner en marcha PEAR y el paquete HTML_BBCodeParser, así que en DesarrolloWeb.com tenemos otras implementaciones de códigos BBCode que se pueden realizar sin necesidad de PEAR: [BBcode con PHP](#) y [Tutorial BBCode](#).

Artículo por Miguel Angel Alvarez

Creación de una gráfica con Google API Chart

Estamos utilizando desde hace poco el sistema de creación de gráficas de Google, que nos ofrece gratuitamente con el servicio API Chart y ha llegado el momento de publicar algunos códigos que venimos utilizando en la creación de gráficos, para que sirvan de guía a los lectores de DesarrolloWeb.com que quieran usar el API de Gráficas en sus webs.

Antes de empezar, para quien no conozca todavía este nuevo servicio de Google, recomendamos la lectura del artículo [API de Google para creación de gráficas](#).

Afortunadamente, Google ha publicado una completa documentación del API Chart en español, por lo que no vamos a explicar los pormenores de la creación de gráficas. En su lugar, os recomendamos que accedáis directamente a la URL donde está la explicación completa del sistema: <http://code.google.com/intl/es/apis/chart/>

En este artículo vamos a mostrar un par de códigos realizados en PHP para crear unas gráficas de tarta y de barras, con datos que extraemos de una base de datos MySQL. Los datos los extraemos de una tabla de respuestas a encuestas y el objetivo es simplemente mostrar gráficos resultado de las votaciones de usuarios a esas encuestas.

Gráfica de tarta

Veamos cómo realizar una gráfica de tarta, también llamada gráfica circular. Como había comentado, los datos los vamos a sacar con PHP de una tabla de respuestas a encuestas, así que nuestro trabajo será obtener los datos de la tabla y generar el código necesario para mostrar la imagen de la gráfica.

Primero vamos a sacar los votos totales de la encuesta. Luego vamos a sacar todas las respuestas que había en la encuesta y vamos a generar una serie de arrays con los datos que utilizaremos para las gráficas.

El siguiente paso será generar la URL de la imagen que tiene que mostrarse para presentar los datos.

```
//votos totales
$sql="select sum(votos) as totales from respuesta where id_encuesta=". $num_encuesta;
```

```
//echo $ssql;
$rs = mysql_query($ssql);
$fila = mysql_fetch_object($rs);
$votos_totales = $fila->totales;
mysql_free_result($rs);

//traigo todas las respuestas
$ssql="select * from respuesta where id_encuesta=". $num_encuesta;
$resultid = mysql_query($ssql);

$colores =
array("e5f867","596605","375181","bfd1d2","e57e0f","54380c","e50f28","a3129e","082454","f6f830","838383");
$texto = array();
$reporte_votos = array();
$nvotos = array();

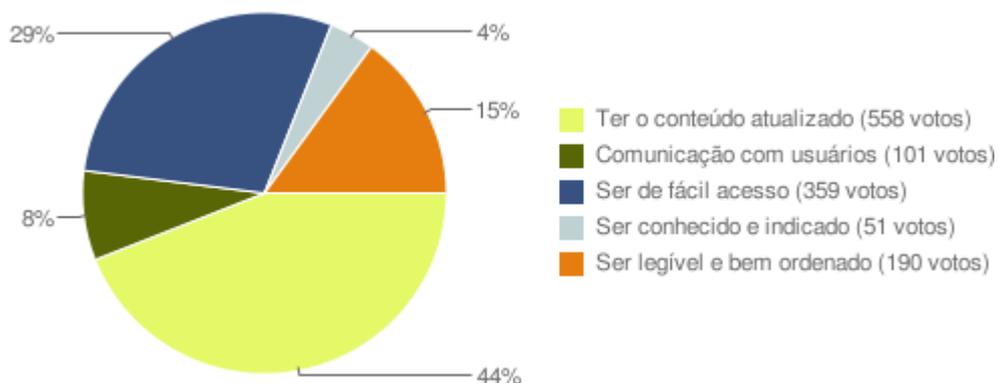
while ( $damefila=mysql_fetch_object($resultid)){
    $porcentaje = round(($damefila->votos/$votos_totales)*100);
    array_push($texto,urlencode(utf8_encode($damefila->respuesta . " (" . $damefila->votos . " votos)"));
    array_push($reporte_votos, urlencode(utf8_encode($porcentaje . "%")));
    array_push($nvotos, $porcentaje);
}

//comienzo la creación de la URL de la gráfica con API Chart
$url_api_chart = "http://chart.apis.google.com/chart?chs=600x200";
//tipo de gráfica de tarta
$url_api_chart .= "&cht=p";

//saco la parte que me interesa del array de colores
$colores_utilizados = array_slice($colores, 0, count($texto));
//colores de la gráfica
$url_api_chart .= "&chco=" . implode($colores_utilizados, ",");
//señales para cada pedazo de tarta
$url_api_chart .= "&chl=" . implode($reporte_votos, "|");
//leyenda
$url_api_chart .= "&chdl=" . implode($texto, "|");
//valores de gráfica
$url_api_chart .= "&chd=t:" . implode($nvotos, ",");

//echo $url_api_chart;
echo "<img src='$url_api_chart' width=400 height=200 border=0>";
```

Una gráfica de tarta que saldría a partir de este ejemplo sería como esta:



Gráfica de barras

Ahora, siguiendo este mismo esquema de programación, vamos a alterarlo un poco para que,

en lugar de una gráfica de tarta, nos muestre una gráfica de barras.

El esquema de acceso a los datos de MySQL continuaría siendo el mismo, aunque ahora se cambiarán algunos detalles de la creación de la gráfica.

```
//votos totales
$ssql="select sum(votos) as totales from respuesta where id_encuesta=". $num_encuesta;
//echo $ssql;
$rs = mysql_query($ssql);
$fila = mysql_fetch_object($rs);
$votos_totales = $fila->totales;
mysql_free_result($rs);

//traigo todas las respuestas
$ssql="select * from respuesta where id_encuesta=". $num_encuesta;
$resultid = mysql_query($ssql);

$colores =
array("e5f867","596605","375181","bfd1d2","e57e0f","54380c","e50f28","a3129e","082454","f6f830","838383");
$texto = array();
$reporte_votos = array();
$nvotos = array();

while ( $damefila=mysql_fetch_object($resultid)){
    $porcentaje = round(($damefila->votos/$votos_totales)*100);
    array_push($texto,urlencode(utf8_encode($damefila->respuesta . " (" . $damefila->votos . " votos)"));
    array_push($reporte_votos, urlencode(utf8_encode($porcentaje . "%")));
    array_push($nvotos, $porcentaje);
}

//comienzo la creación de la URL de la gráfica con API Chart
$url_api_chart = "http://chart.apis.google.com/chart?chs=400x200";
//tipo de gráfica de barras
$url_api_chart .= "&cht=bhg";
//saco la parte que me interesa del array de colores
$colores_utilizados = array_slice($colores, 0, count($texto));
//colores de la gráfica
$url_api_chart .= "&chco=4d89f9|c6d9fd";
$url_api_chart .= "&chco=" . implode($colores_utilizados, "|");
//señales para cada pedazo de tarta
//valores de gráfica
$url_api_chart .= "&chd=t:" . implode($nvotos, ",");
//leyenda
$url_api_chart .= "&chdl=" . implode($texto, "|");

//echo $url_api_chart;
echo "<img src='$url_api_chart' width=400 height=200 border=0>";
```



Con estos ejemplos, y la propia documentación del API, esperamos que los lectores de desarrollo web .com puedan crear sus propias gráficas personalizadas según sus necesidades.

Artículo por Miguel Angel Alvarez

Crear gráficas de mapas con Google API Chart

En este artículo vamos a mostrar como crear mapas, resaltando los países con colores. Para ello tendremos que utilizar un código como este:

```

```

Los parámetros que hemos pasado a la url son los siguientes:

chtm=europe

Tipo de mapa, en este caso es el de Europa, se pueden crear otro tipo de mapas utilizando "africa", "asia", "europe", "middle_east", "south_america", "usa" y "world".

cht=t

Tipo de gráfico, en este caso, mapa.

chs=440x220

Tamaño del mapa, es el máximo permitido.

chco=ffffff,edf0d4,6c9642

Colores para el mapa. El primer color es el que se utiliza por defecto para los países que no pertenecen al parametro chld. El resto de los colores se utilizan para los países incluidos en el parametro chld. El color aplicado al país depende del valor chd.

chld=ES

Países que se van a colorear en el mapa. En este caso solo hemos coloreado el de España. Los códigos que se utilizan en este parámetro son [códigos ISO 3166-1-alpha-2 para países](#), según se define en ISO3166.

chd=s:9

Color para los países especificados en el parámetro chld. En este caso hemos puesto el valor más alto, el 9 que corresponde con el color "6c9642". Para poder poner el color "edf0d4" tendríamos que haber utilizado el valor más bajo, 0. Para cada país del parámetro chld hay que darle un valor en este parámetro. También se puede utilizar valores intermedios.

chf=bg,s,EAF7FE

Relleno del gráfico. En este caso hemos utilizado el color "EAF7FE" para las marcas de agua. "bg" indica que es el relleno del fondo y "s" que el color del relleno es sólido.

Para terminar mostramos como quedaría el mapa.



Creación de un mapa en PHP con datos los datos de una base de datos

Ahora vamos a ver cómo se crearía un mapa del mundo, en un script programado con PHP, resaltando los países del continente Europa. Este script PHP lo hacemos extrayendo los datos necesarios de una base de datos MySQL.

```
//países de Europa
$ssql_map = "select iso_alpha2 from geoname_pais where continent='EU'";
$rs_map = mysql_query($ssql_map);
if(mysql_num_rows($rs_map)!=0){
    while ($fila_map = mysql_fetch_object($rs_map)){
        //códigos de los países de Europa
        $paises.=$fila_map->iso_alpha2;
        //código para el color de cada país
        $codigos.='9';
    }
}
//Imagen del mapa
$imagen_mapa='';

echo $imagen_mapa;
```



Artículo por **Gema Maria Molina Prados**