Unidad 4

Imágenes en PHP

Contacto: consultas@elearning-total.com

Web: www.elearning-total.com

Librería GD

Para muchos desarrolladores web, una herramienta muy útil es la inclusión de la librería para gráficos GD, que permite la manipulación de distintos tipos de imágenes de manera verdaderamente simple y eficaz.

A partir de la versión 4.3.9 de PHP, éste incluye una versión de la librería en su instalación por defecto.

Podemos verificar si la versión de PHP que estamos manejando tiene instala por defecto dicha librería, debemos ver si el fichero php.ini tiene activada esta extensión, para ello basta con utilizar la función phpinfo() que viéramos oportunamente.

Deberíamos ver en el apartado “gd” lo siguiente: GD Support enabled, como vemos a continuación:

gd

GD Support	enabled
GD Version	bundled (2.1.0 compatible)
FreeType Support	enabled
FreeType Linkage	with freetype
FreeType Version	2.8.1
GIF Read Support	enabled
GIF Create Support	enabled
JPEG Support	enabled
libJPEG Version	9 compatible
PNG Support	enabled
libPNG Version	1.6.34
WBMP Support	enabled
XPM Support	enabled
libXpm Version	30512
XBM Support	enabled
WebP Support	enabled

Directive	Local Value	Master Value
gd.jpeg_ignore_warning	1	1

En nuestro caso estamos trabajando con versiones superiores a la 4.3.9 asi que esto debería verificarse siempre.

Funciones GD

Pasemos a continuación a describir algunas de las funciones más comúnmente utilizadas comenzando por aquellas que no tienen un propósito grafico sino más bien de configuración.

Funciones de creación y almacenamiento en archivos

Función	Descripción	Sintaxis
imagecreate	Crea una imagen de dimensiones dadas	$im = imagecreate($x,$y)
imagecreatefromgif	Crea una imagen que tiene como fondo un archivo GIF definido	$im = imagecreatefromgif($archivo)
imagecreatefrompng	Crea una imagen que tiene como fondo un archivo PNG definido	$im = imagecreatefrompng($archivo)
imagecreatefromjpeg	Crea una imagen que tiene como fondo un archivo JPEG definido	$im = imagecreatefromjpeg($archivo)
imagegif	Muestra la imagen creada en el navegador y la guarda como un archivo .gif si se especifica	$im = imagegif($im[,$archivo])
imagepng	Muestra la imagen creada en el navegador y la guarda como un archivo.png si se especifica	$im = imagepng($im[,$archivo])
imagejpeg	Muestra la imagen creada en el navegador y la guarda como un archivo.jpg si se especifica	$im = imagejpeg($im[,$archivo])
imagedestroy	Libera la memoria ocupada por la imagen	imagedestroy($im)

No todas las versiones de bibliotecas GD soportan todos los formatos.

Funciones informativas

Función	Descripición	Sintaxis
getimagesize	Genera un array con las ($arr) informaciones de la imagen: Anchura, altura, formato (1 = GIF, 2 = JPG, 3 = PNG), cadena "height=altura width=anchura" del codigo HTML	$arr = getimagesize ($filename)
imagesx	Devuelve la anchura de la imagen $im	$ancho = imagesx($im)
imagesy	Devuelve la altura de la imagen $im	$alto= imagesy($im)
imagecolorstotal	Devuelve el número total de colores empleados	$total = imagecolorstotal($im)
imagettfbbox	Nos da un array con las coordenadas de las esquinas de un cuadro imaginario que rodea nuestro texto de fuente tipo True Type	$arr = imagettfbbox($talla, $angulo, $archivo_fuente, $texto)

- **getimagesize** puede resultar muy útil para scripts de tratamiento automático de imágenes.
- **imagettfbbox** es muy útil para el centrado y posicionamiento de textos. También puede servirnos en botones dinámicos, para definir el tamaño de la imagen en función del tamaño del texto que vayamos a introducir.

Formatos de imágenes

Pese a que info.php nos devuelve información sobre los tipos de imágenes soportados por la versión en uso de PHP existe una función que permite determinar los formatos soportados.

imagetypes()

Devuelve un campo de bits correspondiente a los formatos soportados por la versión de GD que estamos utilizando.

Los formatos de imagen que actualmente puede soportar PHP son: GIF, JPG, PNG y WBMP.

Veamos a continuación un script que permite obtener información sobre los formatos soportados por la versión de PHP que tenemos instalada.

El conocimiento de estas posibilidades gráficas puede sernos muy útil a la hora de elegir entre los diferentes formatos gráficos disponibles.

```
<?php
if (imagetypes() & IMG_GIF) {
echo "El tipo GIF es soportado<br>";
}else{
echo "El tipo GIF NO ES SOPORTADO<BR>";
}
if (imagetypes() & IMG_PNG) {
echo "El tipo PNG es soportado<br>";
}else{
echo "El tipo PNG NO ES SOPORTADO<BR>";
}
if (imagetypes() & IMG_JPG) {
echo "El tipo JPG es soportado<br>";
}else{
echo "El tipo JPG NO ES SOPORTADO<BR>";
}
if (imagetypes() & IMG_WBMP) {
echo "El tipo WBMP es soportado<br>";
}else{
echo "El tipo WBMP NO ES SOPORTADO ";
} ?>
```

Creación de Thumbnails de imágenes con PHP

Para los que no están familiarizados con el término esto se refiere a previsualizaciones de menor tamaño de una imagen original lo cual se utiliza mucho para galerías, mostrando una copia de la imagen original pero de menor tamaño tanto en pixeles como en kb.

Para el ejemplo utilizaremos como formato de imagen el GIF.

Ejemplo:

```
<?php
$ruta="01.gif";
$fuente = @imagecreatefromgif($ruta);
$alto=200;
$ancho=200;
$imgAncho = imagesx ($fuente);
$imgAlto =imagesy($fuente);
$imagen = ImageCreate($ancho,$alto);
ImageCopyResized($imagen,$fuente,0,0,0,0,$ancho,$alto,$imgAncho,$imgAlt);
imageGif($imagen,"01_thumb.gif");
echo'<img src="01_thumb.gif">';
echo'<br/>';
echo'<img src="01.gif">';
?>
```

Vamos a ver como se hace la previsualización (Thumbnail), como es que se crea desde una imagen más grande una imagen más pequeña tanto en píxeles como en tamaño para su presentación al usuario.

Primero creamos una copia de la imagen y la guardamos en $fuente, esto es necesario ya que será de esta imagen que haremos la previsualización.

```
$fuente = @imagecreatefromgif($ruta);
```

Nota: Recordar que @imagecreatefromgif() es para imágenes gif, si queremos crear un jpeg o png solamente cambiamos el gif: @imagecreatefromjpeg($ruta) o @imagecreatefrompng($ruta)

Ahora obtendremos el ancho y el alto de la imagen original, esto es necesario para poder hacer la copia de la imagen, para ello utilizamos las funciones **imageSX** y **imageSY**, que reciben como

parámetro un identificador de imagen (en este caso $fuente, que es el identificador de la imagen original) y devuelven su ancho y alto.

```
$imgAncho = imagesx ($fuente);
$imgAlto =imagesy($fuente);
```

Ahora, creamos una imagen nueva en blanco con la anchura y altura que queremos para la previsualización (Thumbnail) y que será la que se le devuelva al usuario cuando se le llame, ya sea directamente en el browser o por medio de la etiqueta IMG de html.

```
$imagen = ImageCreate ($ancho,$alto);
```

Ahora lo más importante, copiaremos la imagen original a la imagen nueva, lo cual hará que al tener un menor tamaño (la imagen nueva), la copia de que hacemos de la original se ajustara al tamaño de esta.

Utilizamos la función **imageCopyResized()** la cual sirve para copiar "partes" de una imagen a otra por medio de coordenadas, pero en nuestro caso no necesitamos una parte, necesitamos copiar toda la imagen en todo el espacio de la nueva imagen, por ello damos las coordenadas totales de las imágenes.

```
imageCopyResized ($imagen,$fuente,0,0,0,0,$ancho,$alto,$imgAncho,$imgAlto);
imagecopyresized (resource dst_im, resource src_im, int dstX, int dstY, int srcX, int srcY, int dstW, int dstH, int srcW, int srcH );
```

imagecopyresized() copia una porción rectangular de una imagen hacia otra imagen.

dst_im es la imagen de destino, **src_im** es el identificador de la imagen origen.

Si la altura y anchura de las coordenadas de origen y destino difieren se realizará un estrechamiento o un estiramiento apropiado del fragmento de la imagen.

Las coordenadas van localizadas sobre la esquina superior izquierda. Esta función se puede usar para copiar regiones dentro de la misma imagen (si dst_im es igual que src_im) pero si las regiones se solapan los resultados serán impredecibles.

Listo, ya tenemos nuestra imagen. Ahora guardamos una copia de la imagen para luego mostrarlas (original y copia redimensionada) en el browser (navegador) del usuario.

```
imageGif($imagen,"01_thumb.gif");
```

Al ponerle un segundo parámetro al imageGif, lo que logramos en este caso es que la imagen creada se guarde en la misma carpeta que contiene la imagen original, con el nombre indicado.

Esto puede ser muy útil, ya que si queremos crear thumbnails de todas las imágenes en un directorio, podemos hacer un loop que lea los archivos del directorio, y repita el proceso anterior (de crear imagen en blanco y copiar) y guarde las nuevas imágenes para su uso posterior, realmente es una herramienta con muchos usos.

Thumbnails mejorados

El problema de este sistema de creación de Thumbnails, es que en imágenes de muchos colores, la pérdida de calidad es considerable, afortunadamente, existe una función que permite regenerar prácticamente una imagen con su color real.

Esta función es: imagecreatetruecolor($ancho, $alto);

El resto del proceso ya lo conocemos. En el ejemplo siguiente, mostramos dos thumbnails de la misma imagen realizados con ambas funciones:

```
<?php
$alto=300;
$ancho=400;
$src_img= @imagecreatefromjpeg('space.jpg');
$dst_img = @imagecreatetruecolor($ancho,$alto);
$imagen = @imagecreate($ancho,$alto);
@imagecopyresized($dst_img, $src_img, 0,0,0,0, $ancho, $alto, ImageSX($src_img),
ImageSY($src_img));
@imagejpeg($dst_img,"space_thumb.jpg");
@imagecopyresized ($imagen, $src_img, 0,0,0,0, $ancho, $alto, ImageSX($src_img),
ImageSY($src_img));
@imagejpeg($imagen,"space_thumb2.jpg");
@imagedestroy($dst_img);
echo '<img src="space_thumb.jpg">'; echo '<img src="space_thumb2.jpg">';
?>
```

Guardamos en la variable $src_img una nueva imagen creada de tipo JPEG a partir de space.jpg que será la imagen que redimensionaremos.

Luego en la variable $dest_img, creamos una imagen nueva con color REAL, esta será la que utilizaremos para mostrarla, las variables $ANCHO, $ALTO guardan el nuevo tamaño de la imagen

que obviamente será inferior a la original, por lo cual pueden ayudarse con la función getsizeimage(); que devuelve un vector con los píxeles de X y Y , o en su efecto ImageSX() y ImageSY() para obtener en base a esos parámetros el nuevo tamaño uniforme y acorde con nuestra galería de imágenes.

imagecopyresized(), copia todo o partes de una imagen redimensionada.

Luego mostramos la imagen con **imagejpeg();** si queremos que se guarde en el directorio con sus respectivos permisos, agregaremos un nuevo parámetro:

```
@imagejpeg($dst_img,'NUEVAIMAGEN.JPG');
```

Destruimos la imagen para ahorrar memoria utilizada por imagecreatefromjpeg() y los procesos subsiguientes; y como ya sabemos, el @ (arroba) al comienzo de cada función sirve para evitar que se imprima el error en pantalla.

Marca de Agua

Es muy común ver en la web desarrollos dedicados al tema de proteger el contenido de un sitio web.

Lo mismo pasa con las imágenes, se pueden guardan desde los temporales (suponiendo que por algún extraño motivo no se pueda desde el browser directamente), o simplemente haciendo una captura de pantalla.

Lo que se suele hacer a la hora de proteger imágenes, es aplicarles una marca de agua, es decir, una imagen translúcida que indica que la imagen no puede ser usada en otro sitio, para fines comerciales, etc.

Y es ahí donde la librería GD para tratamiento de imágenes entra en juego. Usando esta librería, se puede automatizar la tediosa tarea de aplicar marcas de agua a diferentes imágenes.

Primero es necesario crear una marca de agua en formato PNG. La ventaja principal de éste formato (indispensable en este caso) es que permite 255 niveles de transparencias, por lo que se puede lograr una imagen translúcida. Una vez creada la marca, se puede aplicar en la imagen usando PHP.

Ejemplo:

```
<?php
$image = "01.gif"; // imagen a aplicar la marca de agua
//$repeat = "t"; // opción para repetir la marca de agua
$watermark = "marca_de_agua.png"; //defnición de la imagen a usar la marca de agua
$im = imagecreatefrompng($watermark);
// crear la marca de agua desde el png
$ext = substr($image, -3); // tomamos la extensión del archivo a proteger
if(strtolower($ext) == "gif") {
//si el archivo es gif creamos la imagen a proteger if (!$im2 = imagecreatefromgif($image))
{
echo "Error opening $image!"; exit;
}
} else if(strtolower($ext) == "jpg") {
//si el archivo es jpgcreamos la imagen a proteger
if (!$im2 = imagecreatefromjpeg($image)) { echo "Error opening $image!"; exit;
}
```

```
} else if(strtolower($ext) == "png") {
//si el archivo es png creamos la imagen a proteger
if (!$im2 = imagecreatefrompng($image)) { echo "Error opening $image!"; exit;
}
} else {
// si la imagen no se correspondió con las 3 opciones de formato, termina el
// script
die;
}
imagecopy($im2, $im, (imagesx($im2)/2)-(imagesx($im)/2), (imagesy($im2)/2)-
(imagesy($im)/2), 0, 0, imagesx($im), imagesy($im));
// copiamos la imagen con la
// marca de agua.
if($repeat) {
// de ser requerido se repite la marca de agua horizontalmente
$waterless = imagesx($im2) - imagesx($im);
$rest = ceil($waterless/imagesx($im)/2); for($n=1; $n<=$rest; $n++) {
imagecopy($im2, $im, ((imagesx($im2)/2)-(imagesx($im)/2))-(imagesx($im)*$n),
(imagesy($im2)/2)-(imagesy($im)/2), 0, 0, imagesx($im), imagesy($im));
imagecopy($im2, $im, ((imagesx($im2)/2)-(imagesx($im)/2))+(imagesx($im)*$n),
(imagesy($im2)/2)-
(imagesy($im)/2), 0, 0, imagesx($im), imagesy($im));
}
}
header("Content-Type: image/jpeg");
imagejpeg($im2);
// se guarda la imagen ahora protegida como jpg imagedestroy($im);
// se borra de memoria la marca de agua utilizada imagedestroy($im2);
// se borra de memoria la imagen utilizada
?>
```