Capítulo 10: Generación dinámica de imágenes.

PHP provee la posibilidad de generar imágenes dinámicamente y de incluir estas imágenes en una página web, esto se hace utilizando funciones de una biblioteca denominada “GD” que viene compilada en forma default en php4, según la versión de GD la biblioteca permita generar imágenes GIF o PNG.
Una vez generada dinámicamente la imagen es posible transmitirla directamente al browser o guardarla en disco para luego levantarla usando un tag <IMG> de html.

Creación de una Imagen.

image_handler=ImageCreate($x,$y);

Crea una Imagen de tamaño X por Y pixels y devuelve un handler a la imagen en $IM (se maneja el handler a la imagen en el resto de las funciones que manipulan la imagen, como si fuera un archivo).

Una vez creada la imagen PHP provee funciones para dibujar rectángulos, arcos, texto y demás elementos en la imagen:

Otras variantes para crear una imagen consisten en crear la imagen a partir de una imagen existente en el disco de forma tal de poder modificarla:

int=imagecreatefromgif(path);
int=imagecreatefrompng(path);
int=imagecreatefromjpg(path);

Al igual que ImageCreate estas funciones devuelven un ImageHandler.

Creación y alocación de colores.

Para utilizar colores en una imagen es necesario en primer lugar crear el color y alocarlo en la imagen, esto se hace con ImageColorAllocate de la siguiente forma:

color_handler=ImageColorAllocate(image_handler,int_rojo,int_verde,int_azul);

La función recibe una image_handler en donde alocar el color y los valores decimales de la cantidad de rojo, verde y azul del color (0 a 255), devuelve un color_handler que puede ser usado en cualquiera de las funciones que veremos a continuación y utilizan colores.

Funciones para creación de objetos en la imagen.

ImageRectangle(image_handler, x1,y1,x2,y2,color_handler);

Dibuja un rectángulo desde la coordenada x1,y1 (0,0 es la esquina superior izquierda de la imagen) hasta la coordenada x2,y2 del color indicado por color_handler (previamente alocado con ImageColorAllocate).

ImageFilledRectangle(image_handler, x1,y1,x2,y2,color_handler);

Idem anterior pero dibuja el rectángulo relleno con el color indicado.

int imagearc (image_handler, cx, cy, ancho, alto, angulo_comienzo, angulo_fin, color_handler)

Dibuja un arco de elipse centrado en cx,cy con el ancho y alto especificado (sin son iguales la elipse es una circunferencia) y desde el ángulo de comienzo al ángulo de fin (en grados 0 a 360). El arco se dibuja con el color indicado.

ImageDashedLine(image_handler, x1,y1,x2,y2,color_handler);

Dibuja una línea puntuada entre las coordenadas especificadas y con el color indicado.

ImageFill(image_handler,x,y,color_handler);

Pinta con el color indicado a partir de la coordenada x,y y con el color indicado, llena con el color indicado.

ImagePolygon(image_handler,array_puntos,cantidad_puntos,color_handler);

Dibuja un polígono usando un vector de puntos de la forma (x0,y0,x1,y1,x2,y2,....etc) el parámetro cantidad_puntos indica cuantos puntos considerar para crear el polígono.

ImageFilledPolygon(image_handler,array_puntos,cantidad_puntos,color_handler);

Idem anterior pero el polígono además se rellena con el color indicado.

ImageLine(image_handler, x1,y1,x2,y2,color_handler);

Dibuja una línea desde x1,y2 hasta x2,y2 con el color indicado.

Manejo de Colores:

color_handler=ImageColorAt(iamge_handler,x,y);

Devuelve el color handler correspondiente al color del pixel especificado.

color_handler=imagecolorclosest (image_handler, int_rojo,int_verde, int_azul)

Devuelve el color alocado más cercano al color indicado en RGB por las cantidades de rojo, verde y azul (en decimal)

color_handler=imagecolorexact (image_handler, int_rojo,int_verde, int_azul)

Idem anterior pero devuelve el color_handler del color pasado si el color no esta alocado en la imagen devuelve –1.

color_handler=imagecolorresolve (image_handler, int_rojo,int_verde, int_azul)

Es una mezcla de las dos anteriores, esta función siempre devuelve un color_handler, o bien el color exacto alocado en la imagen o bien el color más cercano.

int=ImageColorsTotal(image_handler)

Devuelve la cantidad total de colores de la imagen.

imagecolortransparent(image_handler,color_handler)

Setea el color indicado por el handler como transparente para la imagen.

ImageCopy (image_handler_dest,image_handler_origen,x_dest, y_dest, origen_x,origen_y, ancho,alto)

Copia una porción de imagen desde la coordenada origen_x, origen_y con el ancho y alto especificado desde image_handler_origen hacia image_handler_dest en la coordenada x_dest, y_dest.

imagecopyresized (image_handler_dest, image_handler_origen,dest_x,dest_y,origen_x,origen_y,dest_ancho, dest_alto, origen_ancho, origen_alto)

Copia con opción de achicar o agrandar una porción de la imagen hay que especificar la imagen origen, la imagen destino, la coordenada desde donde copiar en la imagen origen, la coordenada a donde copiar en la imagen destino, y el ancho y alto tanto en el origen como en el destino.

Manejo de texto.

imagestring(image_handler,font_number, x, y, string, color_handler)

Coloca un string en la imagen, si font=0 se usa el font default, si font es 1,2,3,4, o 5 se usa un font predefinido.X e Y son las coordenadas donde dibujar el string y especifican la esquina superior izquierda del string.

imagestringup(image_handler,font_number, x, y, string, color_handler)

Idem anterior pero el string se dibuja en forma vertical.

array imagettftext (image_handler,size, angulo,x, y, color_handler, font_path, string)

Dibuja un string en la imagen usando un font treu-type, el font en formato nombre.ttf debe guardarse en algún lugar del file-system que se especifica con font_path (ej.: /fonts/arial.ttf). X e Y son las coordenadas de la esquina inferior izquierda del string. Angulo es el ángulo con el cual se dibuja el string (0=de izquierda a derecha en forma horizontal). Size indica el tamaño en puntos del texto a usar. Devuelve un vector de 8 elementos representando los 4 puntos que delimitan al string de la forma:
izquierda_arriba, derecha_arriba, abajo_izquierda, abajo_derecha. (cada esquina esta representada por dos coordenadas: x e y)

array imagettfbbox (size, ángulo, font_path, string)

Determina el tamaño que ocupara el string en la imagen y devuelve un vector de 8 elementos con el mismo formato descripto en la función anterior.

Generación de la Imagen.

ImageInterlace(image_handler,boolean)

Determina si la imagen será interlazada o no 1=interlazada, 0=no interlazada.

imagegif (image_handler, path)

Dado un image_handler genera un archivo gif con la imagen correspondiente, el archivo puede luego incluirse con un tag <IMG> para mostrarse en una página.

Si no es necesario almacenar la imagen es posible llamar a imagegif de la forma:

imagegif (image_handler)

En cuyo caso la imagen generada es transmitida directamente al browser, para ello antes es necesario enviar al browser un header indicando que se va a recibir un gif.

Ejemplo:

header("Content-Type: image/gif");
ImageGIF($IM);

Ejemplo 1:

```
<?php
Header ("Content-type: image/gif");
$im = imagecreate (400, 30);
$black = ImageColorAllocate ($im, 0, 0, 0);
$white = ImageColorAllocate ($im, 255, 255, 255);
ImageTTFText ($im, 20, 0, 10, 20, $white, "/path/arial.ttf","Testing...
Omega: &#937;");
ImageGif ($im);
ImageDestroy ($im);
?>
```

Este ejemplo genera un gif con un texto usando un font true-type y lo transmite directamente al browser. Es importante que antes y después de los tags <? y ?> no existan espacios en blanco o saltos de línea ya que en ese caso PHP transmitirá al browser estos caracteres y la imagen recibida se vera rota, o sea que debe asegurarse que lo único que recibe el browser es el código de la imagen.

Ejemplo 2:

El siguiente script recibe como parámetros un tamaño x (x), un tamaño y (y), un porcentaje (per), un color de fondo (bg), un color de frente (fg) y dibuja una barra de porcentaje con el porcentaje indicado. Los colores se pasan en hexadecimal. Si no se indica algún parámetro se toman valores default.

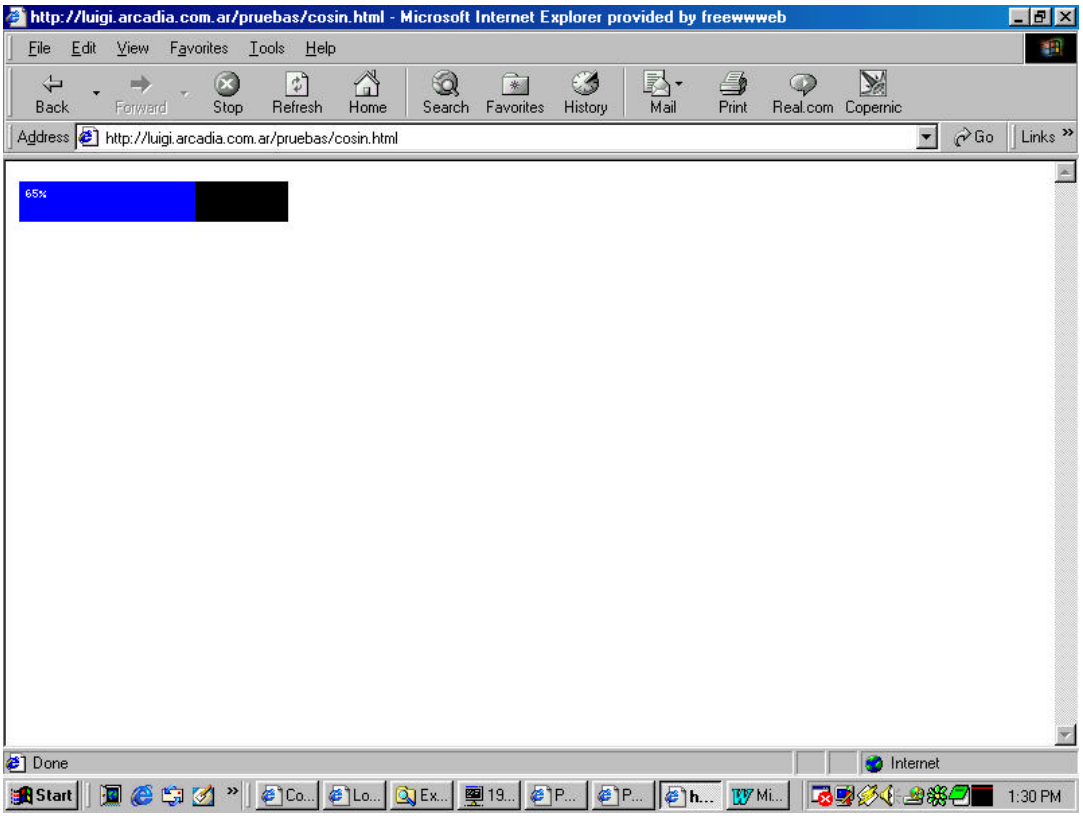
```
<?
//Parámetros:
//Tamaño x, tamaño y, porcentaje de la barra,
//colores ( en notación #FFFFFF)
//En las variables: $x,$y,$per,$bg,$fg

if(!isset($x)){$x=140;}
if(!isset($y)){$y=20;}
if(!isset($per)){$per=75;}
if(!isset($bg)){$bg="#FF0000";}
if(!isset($fg)){$fg="#0000FF";}

$rb=base_convert(substr($bg,1,2),16,10);
$gb=base_convert(substr($bg,3,2),16,10);
$bb=base_convert(substr($bg,5,2),16,10);
$rf=base_convert(substr($fg,1,2),16,10);
$gf=base_convert(substr($fg,3,2),16,10);
$bf=base_convert(substr($fg,5,2),16,10);
$xp=round(($per/100)*$x);

$IM=ImageCreate($x,$y);
$colbg=ImageColorAllocate($IM,$rb,$gb,$bb);
$colfg=ImageColorAllocate($IM,$rf,$gf,$bf);
//$f=ImageLoadFont("fonts/ARIAL.TTF");
$blanco=ImageColorAllocate($IM,255,255,255);
ImageFilledRectangle($IM,0,0,$x,$y,$colbg);
ImageFilledRectangle($IM,0,0,$xp,$y,$colfg);
ImageString($IM,0,5,5,"$per%",$blanco);
```

```
header("Content-Type: image/gif");
ImageGIF($IM);
?>
```

La forma de llamar al script es por ejemplo:

<IMG src="barra.php?x=200&y=30&per=65&bg=#FF4356&fg=#456890">

Y el resultado es:



Como puede verse la llamada es mediante un tag <IMG> con la salvedad de que la imagen que se llama no existe en el disco sino que es un script php que genera dinámicamente la imagen y la entrega al browser.

Utilizando estas funciones de PHP es posible crear gráficos estadísticos en función de datos de la base, modificar imágenes agregando texto, sobreimpresos y realizar varios efectos en función de datos ingresados por el usuario como por ejemplo mapas dinámicos, gráficos de torta, etc...