UTN.BA FACULTAD REGIONAL BUENOS AIRES
SECRETARÍA DE EXTENSIÓN UNIVERSITARIA FRBA UTN
Centro de e-Learning

EXPERTO UNIVERISTARIO EN MySQL Y PHP NIVEL INTERMEDIO

Módulo 2

Mas coceptos sobre Mysql y PHP

Centro de Formación, Investigación y Desarrollo de Soluciones de e-Learning.
UTN - FRBA. Secretaría de Cultura y Extensión Universitaria
Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148 // e-learning@sceu.frba.utn.edu.ar

Librería GD y CAPTCHA.

Centro de Formación, Investigación y Desarrollo de Soluciones de e-Learning.
UTN - FRBA. Secretaría de Cultura y Extensión Universitaria
Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148 // e-learning@sceu.frba.utn.edu.ar

Presentación de la Unidad:

En esta unidad veremos que es la librería GD y su utilidad en la creación de imágenes dinámicas, por ejemplo para crear miniaturas, agregar marcas de agua o generar un CAPTCHA.

También veremos algunos otros ejemplos de CAPTCHA que podamos adaptar a nuestros desarrollos.

Centro de Formación, Investigación y Desarrollo de Soluciones de e-Learning.
UTN - FRBA. Secretaría de Cultura y Extensión Universitaria
Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148 // e-learning@sceu.frba.utn.edu.ar

Objetivos:

- Utilidad de la librería GD en la creación de imágenes dinámicas.
- Uso e implementación de CAPTCHAS.

Centro de Formación, Investigación y Desarrollo de Soluciones de e-Learning.
UTN - FRBA. Secretaría de Cultura y Extensión Universitaria
Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148 // e-learning@sceu.frba.utn.edu.ar

Temario:

Librería GD

CAPTCHA

Centro de Formación, Investigación y Desarrollo de Soluciones de e-Learning.
UTN - FRBA. Secretaría de Cultura y Extensión Universitaria
Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148 // e-learning@sceu.frba.utn.edu.ar

CONSIGNAS PARA EL APRENDIZAJE COLABORATIVO

En esta Unidad los participantes se encontrarán con diferentes tipos de consignas que, en el marco de los fundamentos del MEC*, los referenciarán a tres comunidades de aprendizaje, que pondremos en funcionamiento en esta instancia de formación, a los efectos de aprovecharlas pedagógicamente:

1. Los foros asociados a cada una de las unidades.
2. La Web 2.0.
3. Los contextos de desempeño de los participantes.

Es importante que todos los participantes realicen las actividades sugeridas y compartan en los foros los resultados obtenidos.

Centro de Formación, Investigación y Desarrollo de Soluciones de e-Learning.
UTN - FRBA. Secretaría de Cultura y Extensión Universitaria
Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148 // e-learning@sceu.frba.utn.edu.ar

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 (2.0.34 compatible)
FreeType Support	enabled
FreeType Linkage	with freetype
FreeType Version	2.3.11
T1Lib Support	enabled
GIF Read Support	enabled
GIF Create Support	enabled
JPEG Support	enabled
libJPEG Version	7
PNG Support	enabled
libPNG Version	1.2.40
WBMP Support	enabled
XBM Support	enabled
JIS-mapped Japanese Font Support	enabled

Directive	Local Value	Master Value
gd.jpeg_ignore_warning	0	0

gettext

GetText Support	enabled

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 verdadero propósito grafico sino más bien "administrativo".

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)

A notar que los archivos no tienen por qué ser almacenados como tales. Pueden ser directamente mostrados sin que para ello hayamos ocupado espacio en el disco duro.

Para ello tan sólo hay que omitir la variable *$nombre* (entre corchetes). Por otra parte, no todas las versiones de bibliotecas GD soportan todos los formatos. De hecho, es complicado hacerse con una misma versión que acepte las tres a la vez.

Funciones informativas

Centro de Formación, Investigación y Desarrollo de Soluciones de e-Learning.
UTN - FRBA. Secretaría de Cultura y Extensión Universitaria
Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148 // e-learning@sceu.frba.utn.edu.ar

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. Por otro lado, *imagettfbbox* resulta ser muy práctico para el centrado y posicionamiento de textos al mismo tiempo que 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 tuviéramos 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,$imgAlto);
imageGif($imagen,"01_thumb.gif");
```


Centro de Formación, Investigación y Desarrollo de Soluciones de e-Learning.
UTN - FRBA. Secretaría de Cultura y Extensión Universitaria
Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148 // e-learning@sceu.frba.utn.edu.ar

```
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 el @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.

Ejemplo:

```
<?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(), bueno el nombre lo dice, 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 repetidamente 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. En conclusión, todo lo que llega a la pantalla del usuario le pertenece y puede hacer con eso lo que quiera.

Sin embargo, 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 dicha librería, un poco de PHP y la magia de los PNG's 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
```


Centro de Formación, Investigación y Desarrollo de Soluciones de e-Learning.
UTN - FRBA. Secretaría de Cultura y Extensión Universitaria
Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148 // e-learning@sceu.frba.utn.edu.ar

```
// 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
?>
```

Creación de imágenes dinámicas

Una imagen dinámica es un fichero PHP que contiene las instrucciones para su creación.

Para visualizar una imagen dinámica desde una página web basta con invocar desde la etiqueta clásica de inserción de imágenes de HTML:

```
<img src="nombre del fichero PHP que la genera">.
```

Primera etiqueta

Una vez conocidos los formatos que soporta nuestra versión ya podemos generar imágenes en cualquiera de esos formatos. Vimos un script que ns permite saber esto en la unidad anterior. Para el ejemplo, trabajaremos con dos formatos: JPG y PNG.

Cabe aclarar desde el principio, que en algunas versiones de servidor local, he comprobado problemas al mostrar ciertas imágenes dinámicas en formato PNG, cosa que no sucede una vez colocado on-line el script, sobre todo cuando el PNG no posee color de fondo.

La primera instrucción de un fichero de creación de imágenes dinámicas ha de ser:

header("Content-type: image/jpeg")
si se trata de crear una imagen JPG o:

header("Content-type: image/png")
si pretendemos que la imagen tenga formato PNG.

En el fichero de creación de la imagen no se debe insertar más que el código de creación de la imagen y no se deben dejar líneas en blanco entre la etiqueta de apertura de PHP <?php y la etiqueta header ya que caso contrario nos da un error.

Definida la etiqueta anterior, tenemos que crear la imagen, dibujarla y luego enviarla al navegador para su visualización y por último (no es imprescindible pero si muy conveniente) borrarla con el fin de liberar la memoria ocupada durante su generación.

Estas son las funciones PHP para esos procesos:

$nombre = imagecreate(ancho,alto)

Como ya viéramos en la unidad anterior, con esta función creamos una imagen con el tamaño especificado (ancho y alto) en pixels que es recogida en la variable nombre.

Esta función es idéntica para cualquier formato de imagen.

Envío de imágenes al navegador

Para enviar imágenes al navegador (visualización) se usan funciones diferentes dependiendo del tipo de imagen definida en Header. Si pretendemos que la imagen tenga formato JPG habremos puesto en Header la indicación jpeg (Cuidado que es jpeg y no jpg). En este caso la función de visualización sería:

imagejpeg($nombre)

Si se tratara de una imagen en formato PNG (recuerda que debe estar definido en header) la sintaxis sería:

imagepng($nombre)

Eliminar imágenes de la memoria

Independientemente del formato utilizado para borrar imágenes de la memoria (que no del navegador) se utiliza la siguiente sintaxis:

imagedestroy($nombre)

Como podemos observar en ejemplo_01_imagecreate.php y ejemplo_01_imagecreate.php la imagen que se genera es un cuadrado con fondo negro en el primer caso y un rectángulo con fondo negro en el segundo, ambos con las dimensiones especificadas en cada script.

Ejemplos:

```
<?php
header("Content-type: image/jpeg");
$im = imagecreate(200,200); imagejpeg($im); imagedestroy($im);

?>
<?php
header("Content-type: image/png");
$im = imagecreate(300,200); imagepng($im); imagedestroy($im);
?>
```

Colores

PHP permite crear una paleta de colores. Para ello se pueden crear variables de color (independientemente del formato utilizado) con la siguiente función:

$color=imagecolorallocate ($nombre,R,G,B)

Donde la variable recoge el color especificado por la mezcla de los colores primarios indicados en R, G y B que serán números enteros comprendidos entre 0 y 255 y que especifican la intensidad de las luces roja, verde y azul utilizadas para la obtención del color.

Se pueden definir tantos colores como se deseen sin más que utilizar nombres del variables distintos para cada uno de ellos.

Aplicar colores de fondo

Para aplicar un color de fondo a una imagen (independientemente del formato) se utiliza la siguiente función:

imagefill($nombr,x,y,$col)

Aquí $nombr es la variable en la que se ha creado la imagen, x e y son las coordenadas del punto de la imagen a partir del cual se aplica el relleno y $col el color (previamente definido) aplicable a la imagen.

Vemos a continuación el ejemplo con JPG.
Ejemplo:

```
<?php
header("Content-type: image/jpeg");
$im = imagecreate(200,200);
$fondo=imagecolorallocate ($im, 0, 0, 200);
imagefill ($im, 0, 0, $fondo); imagejpeg($im);

imagedestroy($im);
?>
```

Vemos a continuación el ejemplo con PNG.
Ejemplo:

```
<?php
header("Content-type: image/png");
$im = imagecreate(300,200);
$fondo=imagecolorallocate ($im, 0, 0, 200);
imagefill ($im, 0, 0, $fondo); imagepng($im); imagedestroy($im);
?>
```

Rectángulos sin relleno

Para dibujar un rectángulo sin relleno (solo las líneas) se utiliza la siguiente función: imagerectangle($nom, x0, y0, x1, y1, $col)

Donde $nom es -como siempre- el nombre de la imagen, x0, y0 las coordenadas del vértice superior izquierdo y x1, y1 las coordenadas del vértice inferior derecho y $col el color con el que queremos dibujar las lineas del rectángulo.

Cabe recordar siempre que el punto (0,0) siempre es la esquina superior izquierda de la imagen y que si no usamos para las líneas un color distinto al del fondo no se verá nada.

Ejemplo:

```
<?php
header("Content-type: image/jpeg");
$im = imagecreate(200,200);
$fondo=imagecolorallocate ($im, 0, 0, 200);
$blanco=imagecolorallocate ($im, 255, 255, 255);
imagefill ($im, 0, 0, $fondo);
imagerectangle ($im, 10, 10, 190, 190, $blanco); imagejpeg($im);
imagedestroy($im);
?>
```

Rectángulos con relleno

Para dibujar un rectángulo con relleno se utiliza la siguiente función:

imagefilledrectangle($nom, x0, y0, x1, y1, $col)

Los parámetros son idénticos a los del caso anterior con la única diferencia de que en este caso el rectángulo aparecerá relleno con el color elegido.

Ejemplo:

```
<?php
header("Content-type: image/jpeg");
$im = imagecreate(200,200);
$fondo=imagecolorallocate ($im, 0, 0, 200);
$blanco=imagecolorallocate ($im, 255, 255, 255);
$amarillo=imagecolorallocate ($im, 255, 255,0);
imagefill ($im, 0, 0, $fondo);
imagerectangle ($im, 10, 10, 190, 190, $blanco);
imagefilledrectangle ($im, 20, 20, 180, 180, $amarillo); imagejpeg($im);
imagedestroy($im);
```

```
?>
<?php
header("Content-type: image/jpeg");
$esquinas=array(20,100,100,180,180,100,100,20);
$im = imagecreate(200,200);
$fondo=imagecolorallocate ($im, 0, 0, 200);
$blanco=imagecolorallocate ($im, 255, 255, 255);
$amarillo=imagecolorallocate ($im, 255, 255,0);
imagefill ($im, 0, 0, $fondo);
imagerectangle ($im, 10, 10, 190, 190, $blanco);
imagefilledrectangle ($im, 20, 20, 180, 180, $amarillo); imagefilledpolygon ($im,
$esquinas, 4, $blanco); imagejpeg($im);
imagedestroy($im);
?>
```

Polígonos sin relleno

Su funcionamiento es idéntico al anterior en tanto y cuanto requiere que se defina el array de coordenadas de los vértices y los parámetros de la función son los mismos indicados en el caso anterior.

Solo se modifica el nombre de la función que en este caso es:
imagepolygon($nom, $vert, nº vert , $col)

Ejemplo:

```
<?php
header("Content-type: image/jpeg");
$esquinas=array(20,100,100,180,180,100,100,20);
$im = imagecreate(200,200);
$fondo=imagecolorallocate ($im, 0, 0, 200);
$blanco=imagecolorallocate ($im, 255, 255, 255);
$negro=imagecolorallocate ($im, 0, 0, 0);
$amarillo=imagecolorallocate ($im, 255, 255,0);
imagefill ($im, 0, 0, $fondo);
imagerectangle ($im, 10, 10, 190, 190, $blanco);
imagefilledrectangle ($im, 20, 20, 180, 180, $amarillo); imagepolygon ($im, $esquinas, 4,
$negro); imagejpeg($im);
imagedestroy($im);
?>
```

Elipses, circunferencias y arcos

Una misma función nos permite dibujar elipses, circunferencias y arcos. Es esta:

imagearc($nom, Xc, Yc , a, b, Gi, Gf, $col)

Los parámetros de esta función son los siguientes:
$nom es el nombre de la imagen.

- Xc e Yc son las coordenadas del centro de la elipse.
- a es la longitud del eje horizontal de la elipse.
- b es la longitud del eje vertical de la elipse.
- Gi es el punto inicial del arco expresado en grados sexagesimales.
- Gf es el punto final del arco expresado en grados sexagesimales.
- $col es el color con el que se dibujará la línea.

A los efectos de los ángulos, CERO GRADOS coincide con el cero trigonométrico pero el recorrido de la circunferencia se hace en sentido contrario al trigonométrico es decir, siguiendo el sentido de las agujas del reloj. Obviamente, para dibujar una circunferencia basta con hacer iguales los valores de a y de b.

Ejemplo:

```
<?php
$esquinas=array(20,100,100,180,180,100,100,20);
$im = imagecreate(200,200);
$fondo=imagecolorallocate ($im, 0, 0, 200);
$blanco=imagecolorallocate ($im, 255, 255, 255);
$amarillo=imagecolorallocate ($im, 255, 255,0);
imagefill ($im, 0, 0, $fondo);
imagerectangle ($im, 10, 10, 190, 190, $blanco);
imagefilledrectangle ($im, 20, 20, 180, 180, $amarillo); imagepolygon ($im, $esquinas, 4,
$blanco);
imagearc ($im, 100, 100, 160, 160, 0, 360, $fondo);
imagearc ($im, 100, 100, 160, 100, 0, 360, $rojo); imagejpeg($im);
imagedestroy($im);

?>
```


Centro de Formación, Investigación y Desarrollo de Soluciones de e-Learning.
UTN - FRBA. Secretaría de Cultura y Extensión Universitaria
Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148 // e-learning@sceu.frba.utn.edu.ar

Dibujar sobre una imagen de fondo

PHP permite crear imágenes utilizando como fondo una preexistente. Para ello basta con crear una variable indicando el path y el nombre de la imagen.

Ejemplo:
$b="./images/cruz.png" ó
$b="./images/cruz.jpg"

El formato de esta imagen debe coincidir con el de la que pretendemos construir.

Después de definir esta variable bastaría con sustituir la instrucción de creación de imagen.

$nombre = imagecreate(x,y)
por

$nombre= imagecreatefrompng ($b)
en el caso de imágenes PNG o por

$nombre= imagecreatefromjpeg ($b)
si se tratara de imágenes en formato JPG.

Ejemplo:

```
<?php
header("Content-type: image/jpg");
$esquinas=array(20,100,100,180,180,100,100,20);

$fondo_img="pink.jpg";
$im = imagecreatefromjpeg($fondo_img);
$fondo=imagecolorallocate ($im, 0, 0, 200);
$blanco=imagecolorallocate ($im, 255, 255, 255);
$amarillo=imagecolorallocate ($im, 255, 255,0);
$negro=imagecolorallocate($im, 0, 0,0);
imagefill ($im, 0, 0, $fondo);
imagerectangle ($im, 10, 10, 190, 190, $blanco);
imagefilledrectangle ($im, 20, 20, 180, 180, $amarillo); imagepolygon ($im, $esquinas, 4,
$blanco);
imagearc ($im, 100, 100, 160, 160, 0, 360, $fondo);
imagearc ($im, 100, 100, 160, 100, 0, 360, $rojo); imagejpeg($im);
imagedestroy($im);
?>
```


Centro de Formación, Investigación y Desarrollo de Soluciones de e-Learning.
UTN - FRBA. Secretaría de Cultura y Extensión Universitaria
Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148 // e-learning@sceu.frba.utn.edu.ar

Guardar imágenes

Como vimos en la unidad pasada, las imágenes creadas con la sintaxis anterior no se almacenan como tales en servidor. Si pretendemos que se almacenen deberemos modificar la sintaxis de la etiqueta:

Imagepng($nombre)
o
Imagejpeg($nombre)

añadiéndoles un nuevo parámetro con el nombre y la extensión del fichero gráfico que pretendemos almacenar. Así por ejemplo:

Imagepng($nombre, "mi_imagen.png")
o
Imagejpeg($nombre, "mi_imagen.jpg")

se guardarían en el servidor las imágenes creadas con los nombres mi_imagen.png o mi_imagen.jpg

Trazar segmentos

La función PHP que permite dibujar segmentos rectilíneos es la siguiente:

imageline($nom,x0,y0,x1,y1,$col)

donde:

- $nom es el nombre de la variable definida por imagecreate,
- x0 e y0 son las coordenadas de uno de los extremos;
- x1e y1 son las coordenadas del otro extremo y
- $col es la variable de color con el que será dibujada la línea.

Ejemplo:

```
<?php
1
```

```
header("content-type: image/png");
$im = imagecreate(200,200);
$fondo=imagecolorallocate ($im,0,0,255);
$linea=imagecolorallocate ($im,255,255,255); imageline($im,0,0,200,200,$linea);
imagepng($im);
imagedestroy($im);
?>
```

Colores de fondo

PHP utiliza como fondo de la imagen y de forma automática el primer color definido por la función ImageColorAllocate.

Esta opción de PHP nos obliga a definir dos colores distintos para conseguir la visibilidad de las líneas.

Crear transparencias

Si pretendemos que un determinado color se vuelva transparente debemos utilizar la función:

imagecolortransparent ($nom ,$col).

- $nom es -como siempre- el nombre de la variable definida por imagecreate, y
- $color es el color que pretendemos hacer transparente.

No hay que olvidar nunca dos pequeños detalles:

- Si pretendemos un fondo transparente debemos hacer transparente el primero de los colores definidos.
- Esta función solo tiene sentido para imágenes en formato PNG que admiten transparencias. JPG no las permite.

Ejemplo:

```
<?php
header("content-type: image/png");
$im = imagecreate(200,200);
$fondo=imagecolorallocate ($im,0,0,255);
```

```
$linea=imagecolorallocate ($im,255,0,0); imagecolortransparent ($im ,$fondo);
imageline($im,0,0,200,200,$linea); imagepng($im);
imagedestroy($im);
?>
```

Insertando textos

Estas son las funciones más sencillas que nos permiten insertar textos dentro de una imagen.

imagechar ($im, n, x, y, $txt, $col)

Requiere que la variable $txt contenga una cadena y que esté definida previamente y la utilidad de la función es que inserta el primer carácter de la cadena con orientación horizontal.

Los parámetros de la función son los siguientes:

$nom el nombre de la variable definida por imagecreate

n es un número comprendido entre UNO y CINCO que asigna el tamaño de la letra de menor a mayor.

x e y son las coordenadas del punto donde se colocará la esquina superior izquierda del carácter a representar.

$txt es la cadena de texto de la que se extraerá el primer carácter para ser representado.

$col es el color del carácter a representar.

```
<?php
header("content-type: image/png");
$im = imagecreate(150,150);
$t1="tamaño 1";
$t2="tamaño 2";
$t3="tamaño 3";
$t4="tamaño 4";
$t5="tamaño 5";
$fondo=imagecolorallocate ($im, 0, 0, 200);
$amarillo=imagecolorallocate ($im, 255, 255,0);
imagechar ($im, 1, 0, 0, $t1, $amarillo);
imagechar ($im, 2, 20, 20, $t2, $amarillo);
imagechar ($im, 3, 40, 40, $t2, $amarillo);
```

```
imagechar ($im, 4, 60, 60, $t2, $amarillo);
imagechar ($im, 5, 80, 80, $t2, $amarillo); imagepng($im);
imagedestroy($im);
?>
```

Tenemos además otras funciones para el ingreso de caracteres, por ejemplo:

imagecharup ($im, n, x, y, $txt, $col)

Su funcionamiento es similar a la función anterior con la única diferencia que inserta el carácter con orientación vertical.

Las coordenadas de inserción también se corresponden con la esquina superior izquierda del carácter pero recordemos que ahora estará girado y que por lo tanto ese punto coincidirá con parte inferior izquierda de la imagen del carácter.

Ejemplo:

```
<?php
header("content-type: image/png");
$im = imagecreate(150,150);
$t1="tamaño 1";
$t2="tamaño 2";
$t3="tamaño 3";
$t4="tamaño 4";
$t5="tamaño 5";
$fondo=imagecolorallocate ($im, 0, 0, 200);
$amarillo=imagecolorallocate ($im, 255, 255,0);
imagecharup ($im, 1, 10, 10, $t1, $amarillo);
imagecharup ($im, 2, 20, 20, $t2, $amarillo);
imagecharup ($im, 3, 40, 40, $t2, $amarillo);
imagecharup ($im, 4, 60, 60, $t2, $amarillo);
imagecharup ($im, 5, 80, 80, $t2, $amarillo); imagepng($im);
imagedestroy($im);
?>
```

También podemos utilizar la siguiente función:

imagestring ($im, n, x, y, $txt, $col)

Centro de Formación, Investigación y Desarrollo de Soluciones de e-Learning.
UTN - FRBA. Secretaría de Cultura y Extensión Universitaria
Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148 // e-learning@sceu.frba.utn.edu.ar

Esta función se comporta de forma muy similar a imagechar siendo la única diferencia entre ambas que mientras imagechar inserta solo el primer carácter, en el caso de imagestring se inserta la cadena completa.

Los parámetros de ambas funciones son los mismos.
Si la cadena desborda los límites de la imagen solo se visualizará la parte de la misma contenida dentro de estos.

```
<?php
header("content-type: image/png");
$im = imagecreate(150,150);
$t1="tamaño 1";
$t2="tamaño 2";
$t3="tamaño 3";
$t4="tamaño 4";
$t5="tamaño 5";
$fondo=imagecolorallocate ($im, 0, 0, 200);
$amarillo=imagecolorallocate ($im, 255, 255,0);
imagestring ($im, 1, 10, 20, $t1, $amarillo);
imagestring ($im, 2, 10, 40, $t2, $amarillo);
imagestring ($im, 3, 10, 60, $t3, $amarillo);
imagestring ($im, 4, 10, 80, $t4, $amarillo);
imagestring ($im, 5, 10, 100, $t5, $amarillo); imagepng($im);
imagedestroy($im);
?>
```

La variante vertical sería:
imagestringup ($im, n, x, y, $txt, $col)

No requiere demasiadas explicaciones esta función. Inserta una cadena completa con orientación vertical y sus parámetros son idénticos a los comentados al hablar de imagecharup .

```
<?php
header("content-type: image/png");
$im = imagecreate(150,150);
$t1="tamaño 1";
$t2="tamaño 2";
$t3="tamaño 3";
$t4="tamaño 4";
$t5="tamaño 5";
$fondo=imagecolorallocate ($im, 0, 0, 200);
$amarillo=imagecolorallocate ($im, 255, 255,0);
imagestringup ($im, 1, 10, 100, $t1, $amarillo);
```

```
imagestringup ($im, 2, 20, 100, $t2, $amarillo);
imagestringup ($im, 3, 40, 100, $t3, $amarillo);
imagestringup ($im, 4, 60, 100, $t4, $amarillo);
imagestringup ($im, 5, 80, 100, $t5, $amarillo); imagepng($im);
imagedestroy($im);
?>
```

Tipos de letra

Las funciones anteriores utilizan siempre la fuente predefinida por PHP y solo permiten los cinco tamaños que hemos podido ver en los ejemplos.

Para utilizar tipos de letra y tamaños distintos PHP requiere tener instalada la librería Freetype.

Desde la página oficial de PHP nos remiten a aquí para conseguirla:
/http://www.freetype.org/

CAPTCHA

Captcha es el acrónimo de Completely Automated Public Turing test to tell Computers and Humans Apart (Prueba de Turing pública y automática para diferenciar máquinas y humanos).

Se trata de una prueba desafío-respuesta utilizada en computación para determinar cuándo el usuario es o no humano. El término se empezó a utilizar en el año 2000 por Luis von Ahn, Manuel Blum y Nicholas J. Hopper de la Carnegie Mellon University, y John Langford de IBM.

Prueba de Turing

La típica prueba consiste en que el usuario introduzca un conjunto de caracteres que se muestran en una imagen distorsionada que aparece en pantalla. Se supone que una máquina no es capaz de comprender e introducir la secuencia de forma correcta por lo que solamente el humano podría hacerlo.

Como el test es controlado por una máquina en lugar de un humano como en la Prueba de Turing, también se denomina Prueba de Turing Inversa.

Centro de Formación, Investigación y Desarrollo de Soluciones de e-Learning.
UTN - FRBA. Secretaría de Cultura y Extensión Universitaria
Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148 // e-learning@sceu.frba.utn.edu.ar

El Test de Turing (o Prueba de Turing) es una prueba propuesta por Alan Turing para demostrar la existencia de inteligencia en una máquina. Fue expuesto en 1950 en un artículo (Computing machinery and intelligence) para la revista Mind, y sigue siendo uno de los mejores métodos para los defensores de la Inteligencia Artificial. Se fundamenta en la hipótesis positivista de que, si una máquina se comporta en todos los aspectos como inteligente, entonces debe ser inteligente.

La prueba consiste en un desafío. Se supone un juez situado en una habitación, y una máquina y un ser humano en otras. El juez debe descubrir cuál es el ser humano y cuál es la máquina, estándoles a los dos permitido mentir al contestar por escrito las preguntas que el juez les hiciera. La tesis de Turing es que si ambos jugadores eran suficientemente hábiles, el juez no podría distinguir quién era el ser humano y quién la máquina. Todavía ninguna máquina puede pasar este examen en una experiencia con método científico.

En 1990 se inició un concurso, el Premio Loebner, una competencia de carácter anual entre programas de ordenador que sigue el estándar establecido en la prueba de Turing.

Un juez humano se enfrenta a dos pantallas de ordenador, una de ellas que se encuentra bajo el control de un ordenador, y la otra bajo el control de un humano. El juez plantea preguntas a las dos pantallas y recibe respuestas. El premio está dotado con 100.000 dólares estadounidenses para el programa que pase el test, y un premio de consolación para el mejor programa anual. Todavía no ha sido otorgado el premio principal.

7 pasos para crear un Captcha

Estos son los siete pasos básicos para generar un sistema de Captcha:

1. Generar un código aleatorio
2. Añadir el código a una cookie, variable de sesión ó base de datos que será recuperada desde la siguiente página
3. Escribir el texto dentro de una imagen
4. Mostrar la imagen al usuario que quiera acceder al recurso
5. Proveer de un formulario, donde el usuario ingrese el código y lo envíe
6. Verificar el código de la clave enviada (en el paso 2)
7. Si el código es correcto, se permite el acceso

Generalidades

1

Centro de Formación, Investigación y Desarrollo de Soluciones de e-Learning.
UTN - FRBA. Secretaría de Cultura y Extensión Universitaria
Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148 // e-learning@sceu.frba.utn.edu.ar

Todos estamos cansados de ver imágenes distorsionadas, generalmente con un texto a descifrar e ingresar en ciertos sitios.

En realidad un CAPTCHA es un programa que debería generar una prueba desafío respuesta que los humanos seamos capaces de resolver pero que las máquinas no.

Esto que puede parece algo absolutamente inútil ha cobrado importancia con el auge del spam y el abuso de los recursos ajenos por partes de personas inescrupulosas.

Cuando el spam alcanzó los niveles actuales y la invasión de los correos electrónicos ya no era un desafío para los spammers, los mismos comenzaron a explotar nuevas vías de promoción de forma tal de llevar a los usuarios a los sitios de venta de los productos que publicitan o bien lograr ser indexados mayor cantidad de veces por los buscadores.

La proliferación de blogs y la posibilidad de enviar mensajes a los mismos, sin registración y con formularios que no requieren más que el ingreso de texto proporcionó a los spammers la forma perfecta de lograr sus objetivos.

Con el desarrollo de simples bots (Un bot (abreviatura de robot) es un programa informático que realiza funciones muy diversas, imitando el comportamiento de un humano), se rastrean millones de blogs enviando direcciones de los sitios web promocionados por los spammers. Esta acción se conoce como Splog (Un splog es un blog creado con el único fin de promocionar sitios web afiliados, mejorar la posición de los mismos ante los buscadores, para publicar únicamente avisos publicitarios que pagan por cantidad de visitantes. El contenido de estos blogs es generalmente texto escrito específicamente para obtener mayores ganancias con AdSense o también textos copiados desde otros sitios web. El término splog es un neologismo que viene de contraer la expresión spam blog del idioma inglés.).

Como se mencionó anteriormente la ventaja radica en dos puntos importantes para el spammer:

Si el blog es importante y muy visitado, miles de usuarios podrán ver los enlaces y hacer click en ellos.

Si los buscadores indexan este enlace, el mismo puede ser considerado de relevancia directamente proporcional a la importancia del blog del que fue obtenido. Si bien los buscadores implementan algoritmos (y tags HTML) para evitar estos sitios, los mismos son burlados y es común ver sitios de spammers con los buscadores.

Esta última metodología también puede ser aplicada a sitios tan populares como la Wikipedia si bien se implementan métodos para evitarlos, como el tag "noindex" que le dice a los buscadores que no indexen los enlaces contenidos en la página.

Los CAPTCHAs han sido, por un tiempo, una aproximación bastante acertada a la solución del problema hasta que los spammers encontraron la forma de saltar algunos de ellos.

La forma de pasar un CAPTCHA es el reconocimiento óptico de caracteres (OCR), la misma técnica actualmente utilizada desde hace años en los scanners y la misma utilizada para detectar spam gráfico en los correos electrónicos.

Es así que actualmente pueden verse decenas de CAPTCHA distintos, algunos de los cuales suelen ir desde soluciones muy sencillas, por lo cual no cumplen su función, a extremadamente difíciles de descifrar incluso para humanos. Esto sin mencionar que las personas con deficiencias visuales tienen problemas con su reconocimiento.

Es importante remarcar que la "fortaleza" de un CAPTCHA puede ser distinta según la utilización del mismo: no tiene la misma importancia el registro a un sitio web como paypal que un formulario de contacto en un weblog.

A modo de ejemplo podemos citar proyectos en los cuales el objetivo es mejorar los CAPTCHAs actuales, mostrando debilidades, fortalezas, eficacia e ineficiencia de muchos de los métodos utilizados.

Otras corrientes prefieren variar estos métodos mediante la utilización de algunos de los siguientes métodos:

Resolver un problema del tipo matemático - ¿Cuánto es siete menos cuatro?
Presenta problemas con el idioma y por alguna razón es rechazado por muchos usuarios.

Cuál de estas personas es mayor? y se muestran un bebé y una persona adulta.
Presenta problemas en las personas con deficiencias visuales.

Responder preguntas del tipo ¿cuál es la temática de este sitio?
Pueden presentar problemas con el idioma o con la interpretación de la pregunta realizada.

Establecer la relación que guardan ciertas imágenes. Mostrar cuatro imágenes de gatos y la persona debe seleccionar "gato" desde una lista desplegable.
Presenta los mismos problemas del anterior y además algunas imágenes pueden ser difíciles de relacionar.

Establecer relación entre imágenes y caracteres en un captcha complejo.
Pueden presentar los problemas de los anteriores e incluso llegar a ser muy complejos.

Resolución de Puzzles y rompecabezas.
Pueden ser complejos. Presenta problemas en las personas con deficiencias visuales.

Crear CAPTCHAs auditivos.
Presentan problemas con las personas con discapacidades auditivas.

Centro de Formación, Investigación y Desarrollo de Soluciones de e-Learning.
UTN - FRBA. Secretaría de Cultura y Extensión Universitaria
Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148 // e-learning@sceu.frba.utn.edu.ar

Si bien las alternativas son muchas, algunos profesionales afirman que este no es el camino correcto debido a los problemas de accesibilidad que presentan y que cualquier método técnico a la larga es "hackeable" por un robot (o por una máquina que pase el Test de Turing).

Más allá de no ser la solución final, lo cierto es que actualmente son un filtro importante en la mayoría de los sitios webs y todo parece indicar que los CAPTCHAs nos acompañaran por un tiempo más hasta que se perfeccionen algunos métodos de detección heurísticos.

Pasemos a un ejemplo sencillo
Lo primero que necesitamos para mostrar un CAPTCHA es crear aleatoriamente la secuencia de caracteres que va aparecer en la imagen y guardarla en una variable de sesión ($_SESSION) para comprobarla luego con la ingresada por el usuario y un formulario para que el usuario ingrese la clave.

```
<?php
session_start(); // Iniciamos la sesión

?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Documento sin t&iacute;tulo</title>
</head>
<body>
<?php
$numero1= rand(0,9);
$numero2= rand(0,9);
$numero3= rand(0,9);
$minus =
array("a","b","c","d","e","f","g","h","i","j","k","l","m","n","ñ","o","p","q","r","s","t","
u"
,"v","w","x","y","z");
$mayus =
array("A","B","C","D","E","F","G","H","I","J","K","L","M","N","Ñ","O","P","Q","R",
"S","T","U","V","W","X","Y","Z");
$signos = array("!","#","$","%","&","=");
$generador_min = rand(0,26);
$generador_may = rand(0,26);
$generador_sig = rand(0,5);
$_SESSION["codigos"] =
($numero1).($minus[$generador_min]).($numero2).($mayus[$generador_may]).
($signos[$generador_sig]).($numero3); // Guardamos el texto del CAPTCHA en la sesion
```


Centro de Formación, Investigación y Desarrollo de Soluciones de e-Learning.
UTN - FRBA. Secretaría de Cultura y Extensión Universitaria
Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148 // e-learning@sceu.frba.utn.edu.ar

```
print "<img src=imagen2.php>";
?>
<form action="confirmacion2.php" method="post"> Escriba codigo de seguridad:
<label>
<input name="confirmacion" type="text" id="confirmacion" />
</label>
<p>
<label>
<input type="submit" name="Submit" value="Comprobar" />
</label>
</p>

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

Dentro del script anterior, encontramos una llamada a una archivo php que genera dinámicamente la imagen tal cual viéramos en las unidades anteriores.

Lo trascribimos a continuación:

```
<?php session_start();
header ("Content-type: image/jpeg");
$im = @imagecreate(100, 30);
$color_fondo = imagecolorallocate ($im, 240, 240, 240);
$color_texto = imagecolorallocate ($im, 0, 128, 6);
imagestring ($im, 25, 25, 5, $_SESSION["codigos"], $color_texto); imagejpeg ($im);
?>
```

Por último, lo que necesitamos es hacer la comparación entre la clave ingresada en el formulario y la generada automáticamente.

```
<?php session_start();
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Documento sin t&iacute;tulo</title>
</head>
```


Centro de Formación, Investigación y Desarrollo de Soluciones de e-Learning.
UTN - FRBA. Secretaría de Cultura y Extensión Universitaria
Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148 // e-learning@sceu.frba.utn.edu.ar

```
<body>
<?php
$confirmacion = $_POST[confirmacion]; if($confirmacion == $_SESSION['codigos'])
{
print "Codigo correcto";
}else{
print "error";
}

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

Afortunadamente tenemos a disposición varios CAPTCHAs, veremos a continuación varios de ellos.

Captcha Fácil es un pequeño programa realizado en PHP que se utiliza como si fuera una imagen dentro de un formulario. Al insertar la imagen en el formulario, el programa guardará en la sesión del usuario la secuencia de letras y números que corresponde al

Captcha.
Cuando procesamos el formulario en el paso siguiente, simplemente deberemos verificar que el usuario haya ingresado en el campo de texto el mismo valor que está guardado en la sesión.

Formulario

```
<html>
<head>
</head>
<body>
<form action="verificacion.php" method=”post”>
<img src="captcha.php" /><br/>
<input type="text" size="16" name="captcha" />
<br/><br/>
<input type="submit" />
</form>
</body>
</html>
```

Comprobación

```
<?php session_start();
if(strtoupper($_POST["captcha"]) == $_SESSION["captcha"]){
```

```
// REMPLAZO EL CAPTCHA USADO POR UN TEXTO LARGO PARA EVITAR QUE SE VUELVA A
INTENTAR
$_SESSION["captcha"] = md5(rand()*time());
// INSERTA EL CÓDIGO EXITOSO AQUI
echo "aprobado";
}else{

// REMPLAZO EL CAPTCHA USADO POR UN TEXTO LARGO PARA EVITAR QUE SE VUELVA A
INTENTAR
$_SESSION["captcha"] = md5(rand()*time());
// INSERTA EL CÓDIGO DE ERROR AQUÍ
echo "reprobado";
}
?>
```

Generación de la imagen

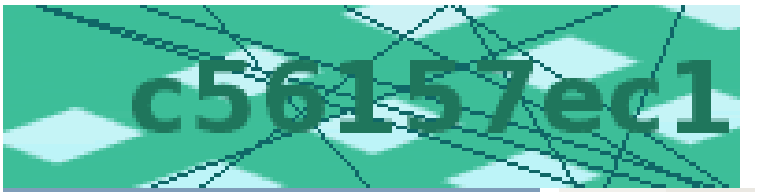
```
<?php
#crear la imagen y el color de fondo
$captcha = imagecreatetruecolor(120,35);
$color = rand(128,160);
$background_color = imagecolorallocate($captcha, $color, $color, $color);
imagefill($captcha, 0, 0, $background_color);
#dibujamos algunas líneas
for($i=0;$i<10;$i++){
$color = rand(48,96);
imageline($captcha, rand(0,130),rand(0,35), rand(0,130), rand(0,35),imagecolorallocate
($captcha, $color, $color, $color));
}
#generamos un valor aleatorio de 5 dígitos
$string = substr(md5(rand()*time()),0,5);
#convertimos el string en uppercase y reemplazamos "O" y "0" para evitar errores
$string = strtoupper($string);
$string = str_replace("O","B", $string);
$string = str_replace("0","C", $string);
#grabamos el string en la session "captcha"
session_start();
$_SESSION["captcha"]=$string;
#ubicamos los elementos del random
$font = 'arial.ttf'; for($i=0;$i<5;$i++){
$color = rand(0,32); if(file_exists($font)){
$x=4+$i*23+rand(0,6);
$y=rand(18,28);
imagettftext ($captcha, 15, rand(-25,25), $x, $y, imagecolorallocate($captcha,
$color,
```

```
$color, $color), $font, $string[$i]);
}else{
$x=5+$i*24+rand(0,6);
$y=rand(1,18);
imagestring($captcha, 5, $x, $y, $string[$i], imagecolorallocate($captcha, $color,
$color, $color));
}
}
#distorcionamos la imagen
$matrix = array(array(1, 1, 1), array(1.0, 7, 1.0), array(1, 1, 1));
imageconvolution($captcha, $matrix, 16, 32);
header("Expires: 0");
header("Cache-Control: must-revalidate, post-check=0, pre-check=0"); header("Cache-
Control: private",false);
#convertimos a imagen header("Content-type: image/gif"); imagejpeg($captcha);
?>
```

Fuzzy CAPTCHA

Esta es la traducción de un artículo publicado en fuzzyopinions.

El resultado final del Captcha lo pueden observar en la siguiente imagen:



Cómo dijimos, un CAPTCHA es solicitar al usuario que escriba los caracteres que aparecen en una imagen dentro de un campo de texto. Si lo hace correctamente tendrá acceso, de lo contrario, se le pedirá que vuelva a escribir el código, indicándole que lo que ha escrito esta errado.

A continuación veremos el código necesario para crear nuestro CAPTCHA

Generando y almacenando el código

Cómo mencionamos en el primer paso, el texto del código debe ser aleatorio. Hay varias maneras de conseguir esto, por lo que en este paso podemos sentirnos libres de experimentar, pero eso si, lo que hagamos debe de cumplir dos cosas: 1) Recuperar un valor que cambie constantemente y 2) Cifrar ese valor.

Un valor que cambia constantemente puede ser, por ejemplo, la fecha y hora del servidor, para ello, nos serviremos de las funciones microtime( ) y mktime( ) de PHP.

Una vez con el valor totalmente aleatorio en nuestras manos, lo cifraremos con la función md5() de PHP que veremos en detalle en el curso avanzado. Finalmente (fíjense en la última línea del código), almacenaremos el valor en una variable de sesión.

```
<?php
// Iniciamos sesión
session_start( );
// Indicamos el tamaño de nuestro captcha, puede ser aleatorio para mayor seguridad
$captchaTextSize = 7;
do {
// Generamos un string aleatorio y lo encriptamos con md5
$md5Hash = md5( microtime( ) * mktime( ) );
// Eliminamos cualquier caracter extraño
preg_replace( „([1aeilou0])“, “”, $md5Hash );
} while( strlen( $md5Hash ) < $captchaTextSize );
// necesitamos sólo 7 caracteres para este captcha
$key = substr( $md5Hash, 0, $captchaTextSize );
// Guardamos la clave en la variable de sesión. La clave esta encriptada.
$_SESSION[“key“] = md5( $key );
?>
```

Mostrando el código aleatorio en la imagen Captcha

Una vez generado el código captcha y almacenado correctamente, procederemos a recuperarlo y mostrarlo al usuario. Para el manejo de la imagen utilizaremos la librería GD que ya hemos visto en las unidades pasadas.

```
<?php
// almacenamos la imagen base, el background.
$captchaImage = imagecreatefrompng( “images/captcha.png” );
```

```
/* Seleccionamos un color de texto. Cómo nuestro fondo es un verde agua, escogeremos un cólor verde para el texto. El color del texto es, preferentemente, el mismo que el del background, aunque un poco más oscuro para poder distnguirlo. */
$textColor = imagecolorallocate( $captchaImage, 31, 118, 92 );
/* Seleccionamos un color para las líneas que queremos se dibujen en nuestro captcha. En este caso usaremos una mezcla entre verde y azul */
$lineColor = imagecolorallocate( $captchaImage, 15, 103, 103 );
?>
```

Ya hemos seleccionado los colores, ahora queremos mostrar algunas líneas que incrementen la dificultad, para que los robots (spam) no lean nuestro código (texto aleatorio).

```
<?php
// recuperamos el parametro tamaño de imagen
$imageInfo = getimagesize( “images/captcha.png” );
// decidimos cuantas líneas queremos dibujar
$linesToDraw = 10;
// Añadimos las líneas de manera aleatoria
for( $i = 0; $i < $linesToDraw; $i++ ) {
// utilizamos la función mt_rand()
$xStart = mt_rand( 0, $imageInfo[ 0 ] );
$xEnd = mt_rand( 0, $imageInfo[ 0 ] );
// Dibujamos la linea en el captcha
imageline( $captchaImage, $xStart, 0, $xEnd, $imageInfo[1], $lineColor );
}
?>
```

Finalmente, escribimos nuestro código en la imagen y la mostramos en la pantalla. Comenzaremos con una imagen de fondo pre generada. Este método, puede fácilmente permitirnos incluir docenas de fondos adicionales, seleccionados al azar, o aún, para hacerlo más complejo, podríamos generarlo nosotros mismos, dentro del captcha.

Para escribir nuestro captcha usaremos la función imagettftext(), la cual se caracteriza por escribir un texto sobre una imagen usando fuentes TrueType. En este caso, utilizaremos la fuente BitStream Vera Sans Bold, que es una fuente Open Source que podemos copiar y redistribuir libremente. Ahora bien, si queremos podemos utilizar cualquier otra fuente, ó aún mejor, mezclar fuentes al azar para hacer más difícil que los robots lean nuestra imagen.

```
<?php
/* Escribimos nuestro string aleatoriamente, utilizando una fuente true type. En este caso, estamos utilizando BitStream Vera Sans Bold, pero podemos utilizar cualquier otra. */
imagettftext( $captchaImage, 20, 0, 35, 35, $textColor, “fonts/VeraBd.ttf”, $key );
```

```
/* Mostramos nuestra imagen. Preparamos las cabeceras de la imagen previniendo que no se almacenen en la cache del navegado */
header ( “Content-type: image/png” ); header(”Cache-Control: no-cache, must-revalidate”);

header(”Expires: Fri, 19 Jan 1994 05:00:00 GMT”); header(”Pragma: no-cache”);
imagepng( $captchaImage );
?>
```

Ahora que ya tenemos nuestro captcha elaborado, el paso final es crear el formulario que interactúe con el usuario.

Incluyendo el Captcha en la página web

Para verificar el correcto funcionamiento de nuestro captcha, necesitamos crear un formulario que permita al usuario ver el código, escribirlo en un campo de texto y enviarlo pulsando un botón. Esto ya todos lo debemos de conocer, por lo que no nos explayamos en este punto.

```
<img src=”captcha.php” border=”0” />
<form name=”captcha-form” method=”POST” action=”captcha-verify.php”>
<input type=”text” name=”code” width=”25” />
<input type=”submit” name=”submit” value=”submit” />
</form>
```

Ahora bien, asumiendo que el formulario ha sido enviado, deberemos ser capaces de verificar el código enviado. Por ejemplo podemos utilizar un script PHP similar a este.

```
<?php
session_start( ); // iniciamos sesión
/* Encriptamos la clave pasada por el formulario y luego la comparamos con el valor del captcha (almacenado en la variable de sesión) */
if( md5( $_POST[ „code‟ ] ) != $_SESSION[ „key‟ ] ) {
echo “Usted no ha escrito el código de verificación correctamente. Por favor, inténtelo de nuevo!”;
} else {
echo "Usted ha escrito el código correctamente. Bienvenido...";
```

```
}
?>
```

Eso es todo. Desde luego, hay mucho que se podría hacer para personalizar un Sistema Captcha, hacerlo más complejo e incluirlo en nuestros sitios web. Estas son algunas ideas:

- Combinar palabras para hacerlo semilegible y dificultar a los robots que lean la imagen. Esto se podría hacer utilizando dos archivos de donde se recuperen los textos.
- Girar el texto al azar y jugar con el espacio entre las letras. Aunque hay que tener cuidado de no hacerlo muy enrevesado, tanto que nuestros propios usuarios no puedan entenderlo.
- Cambie la ubicación del texto en la pantalla. En nuestro caso, el texto siempre aparece en un mismo punto, pero podríamos cambiar su ubicación al azar.

Secureimage, un captcha en PHP open source

Uno de los filtros antispam más comunes se conoce como Captcha y consiste en una imagen con una palabra escrita que el usuario humano debe escribir en un cuadro de texto para así ponérselo difícil a los bots de SPAM. El proceso resulta molesto para el usuario pero puede ayudar a filtrar el spam y en algunos casos incluso a evitar una oleada que casi tumbe tu servidor.

Si estás pensando en colocar uno de estos módulos en tu web te aviso de que existen opciones de código abierto (open source) como Secureimage, un captcha en PHP open source que puedes descargarte en su página oficial:

http://www.phpcaptcha.org/download

Lo cierto es que es un sistema que se puede integrar en pocos minutos y con bastante facilidad, dado que tiene una documentación completa que ofrece desde su propia página web, además de un minitutorial para comenzar a usar el sistema rápidamente.

Claro que la documentación está en inglés, con lo que la vamos a traducir libremente en este artículo y de paso ampliarla con algún detalle adicional.

Qué es Securimage

Securimage es un script de código abierto, gratuito y libre de uso, que sirve para integrar una Captcha en sistemas PHP. Un captcha como ya vimos, es una imagen de confirmación humana, que sirve para saber que es una persona la que está escribiendo un formulario y no una máquina. Seguro que habremos visto cientos de captchas en formularios de páginas web, que nos obligan a escribir el texto de una imagen en un campo, que tiene que coincidir con lo que está dibujado, para que el formulario se envíe correctamente.

Requisitos para instalar esta Captcha PHP
Los requisitos para instalarla son tener PHP (son válidas las versiones 4 ó 5 de PHP) y la librería GD. Eso es lo básico, que casi todos los sistemas actualmente lo tienen, puesto que a partir de determinada versión de PHP 4, las librerías GD están instaladas de manera predeterminada.

Para quien no sepa que es la librería GD hay que decir que son un juego de funciones para manipulación de imágenes, es decir, para la creación de ficheros gráficos en varios formatos, desde PHP y de manera dinámica. Nosotros hemos hablado sobre GD en otras unidades.

Además hay otro requisito opcional, para que funcione Securimage con todas sus funcionalidades. Es tener instalado FreeType, que es un motor para trabajo con fuentes TTF. Este requisito es opcional, porque puede funcionar sin él, aunque no utilizará la parte de fuentes True Type, para configurar el texto de la imagen con distintas fuentes.

En este caso simplemente se generará siempre con el mismo tipo de fuente muy fea ella, pero legible.

Aparte necesitaremos que nuestro PHP tenga soporte para sesiones, algo totalmente básico que tienen todos los PHP por lo general.

Para instalar éste Captcha open source en tu web basta con que descomprimas el archivo en el directorio de la misma e introduzcas las siguientes líneas de código:

Primero, pon ésta línea de código en donde quieras que se muestre la imagen con la palabra. Cuidado con la ruta de la imagen, ha de coincidir con la ruta en que se encuentra ese archivo en tu web.

```
<img id="captcha" src="/securimage/securimage_show.php" alt="CAPTCHA Image"
/>
```

Segundo, pon ésta otra línea de código que será el cuadro en que el usuario deberá escribir la palabra. Ojo con la propiedad name que la necesitaremos luego.

```
<input type="text" name="captcha_code" size="10" maxlength="6" />
```

Opcionalmente tenemos la posibilidad de habilitar la opción de recargar imagen por si el usuario es incapaz de reconocer la imagen generada automáticamente. Fíjate en que la id de la imagen

anterior ha de coincidir con lo que apunta el getElementById y en que nuevamente la nueva ruta de la imagen debe coincidir con el archivo en tu web.

```
<a href="#" onclick="document.getElementById('captcha').src = '/securimage/securimage_show.php?' + Math.random(); return false">Cargar otra imagen.</a>
```

Ahora que ya tenemos preparada la interfaz que utilizará el usuario, debemos colocar un código en el servidor que recogerá lo enviado por el usuario y analizará si es o no correcto.

Hay varias formas de hacer esto, yo he empleado un postback sobre la misma página, por tanto, establezco un condicional que en caso de estar realizando un postback y ya que en esa página es el único formulario posible comprueba que todo sea correcto.

Dentro del postback hay que colocar las siguientes instrucciones. Ojo con session_start(); que debe ejecutarse antes de que se haya escrito ningún HTML, lo mejor es colocarlo a principio de toda la página pero puedes ponerlo más abajo siempre que te asegures de que no se ha escrito nada de HTML aún.

```
if( strtoupper( $_SERVER['REQUEST_METHOD'] ) == "POST" ) { // Es un postback
session_start();
include './ModCaptcha/securimage.php';
$securimage = new Securimage();
if ($securimage->check($_POST['captcha_code']) == false) {
// El código no es correcto
} else {
// Envío email
}
}
```

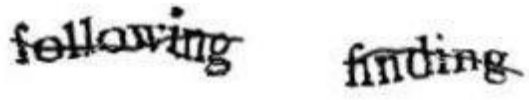
Como ves, tras crear y cargar el objeto Secureimage comprobamos si el código introducido por el usuario es correcto, a partir de aquí te toca a ti realizar las distintas operaciones para mostrar el error o enviar el mensaje etc.

ReCAPTCHA

Puede descargarse desde:
http://www.google.com/recaptcha

reCAPTCHA es una extensión de la prueba CAPTCHA que se utiliza para reconocer texto presente en imágenes. Emplea por tanto la prueba desafío-respuesta utilizada en computación para determinar cuándo el usuario es o no humano para, a su vez, mejorar la digitalización de textos.

reCAPTCHA se basa en el hecho de que para un ser humano puede ser simple determinar el texto presente en una imagen cuando para una máquina esta tarea resulta en ocasiones demasiado compleja.



Ejemplo de una prueba de reCAPTCHA que contiene las palabras del idioma inglés following finding.

El reCAPTCHA trata de solucionar un problema de partida: cuando se digitaliza un documento impreso se toman fotografías del mismo y esas fotografías se convierten a texto empleando sistemas OCR. Sin embargo, ocurre que hay palabras que presentan dificultades para ser reconocidas automáticamente: aquellas que contienen letras deformes, manchas producto de defectos en la impresión del papel, páginas con polvo, entre otras. Estas palabras pueden ser identificadas por personas de manera mucho más confiable que por un sistema OCR comptuarizado. reCAPTCHA emplea esta facilidad del ser humano, para así lograr un método de reconocimiento de texto mucho más confiable.

El uso de reCAPTCHA consiste en sustituir al sistema CAPTCHA, colocando dos palabras a reconocer (en lugar de una que emplea típicamente la prueba CAPTCHA).

Una de las palabras es conocida y la otra es desconocida para el sistema. La palabra desconocida es una que no pudo ser obtenida de una imagen mediante un sistema OCR automatizado. El

sistema pide al usuario (quien desconoce qué palabra es conocida y cuál no lo es) que introduzca ambas palabras como texto. Si la palabra conocida por el sistema es introducida correctamente por un humano, el sistema reCAPTCHA asume que hay probabilidades altas de que el usuario también haya introducido la palabra desconocida correctamente. Si la palabra desconocida recibe en múltiples ocasiones la misma transcripción humana (traducción de imagen a texto) se considera que esa transcripción es correcta. De esta forma, a la prueba desafío-respuesta utilizada en computación para determinar cuándo el usuario es o no humano (prueba CAPTCHA) se le suma la utilidad de permitir mejorar la digitalización de textos.

Las palabras que fueron traducidas en muchas ocasiones de la misma manera, se pueden incorporar como palabras conocidas dentro del propio sistema.

Actualmente reCAPTCHA es utilizado para digitalizar ediciones impresas del New York Times. La compañía dueña del sistema reCAPTCHA fue adquirida por Google que podrá usar el sistema como apoyo para su proyecto Google Books.