Roadmap Series

Issue 9

Dynamic Image Watermarks using PHP

Prepared by Justin Nemeth

5-28-09

Recommended Experience Level

1 Beginner 2 Intermediate [✓ 3 Advanced]

Dynamic Image Watermarks using PHP

Roadmap Series

Issue 9

Dynamic Image Watermarks using PHP

Produced by Justin Nemeth

PHP can do a lot more things than just generate dynamic HTML or retrieve records from a database. Using the built-in GD image library, PHP can dynamically manipulate and create images for you. Some examples of what you can do include.

- overlay a text watermark
- create graphical buttons with dynamic text
- use external fonts
- resize, crop, or create thumbnails
- convert from one image type to another

As you can see, the GD library is very powerful and allows you to manipulate images in several ways. For the purpose of this Roadmap, we will focus on creating a PHP script that will overlay a text watermark onto an existing image.

Is the GD Library enabled?

In a default PHP installation, the GD library should already be enabled. If you are unsure, or if you are experiencing problems, review the following link for instructions on how you can turn on the GD library.

www.webassist.com/go/howto/php_gd

Load an existing image

As a starting point, we want to load an existing image which we will later overlay a text watermark on. For this example we are going to apply this watermark to a jpeg file, *example.jpg*.

In Dreamweaver, create a new blank PHP page and save it as *image.php* to a folder of your choice. In the same folder, place the *example.jpg* image. This can be any jpeg image file, just replace *example.jpg* with your filename.

Inside the file, delete all text and just put `<?php ?>` in the code. Next, copy and paste the following line of code to load *example.jpg* and give us an image resource. We will manipulate this image resource with our watermark and then output to the browser in the later steps.

```
//load image
$im = imagecreatefromjpeg('example.jpg');
```

Determine Watermark Location

The watermark will be placed in the bottom left corner of the image, so we need to do some basic calculations to figure out the x and y coordinates for the text to start at. The top left corner is *x=0* and *y=0*. So, we want a slight x offset and a y offset of almost the entire image height. Add the following code.

```
//figure out where to put the text
$imagesize = getimagesize('example.jpg');
```

```
$x_offset = 7;
$y_offset = $imagesize[1] - 20;
```

In the above code, we first get some size related info for the *example.jpg* image. The getimagesize function returns a numeric array of information. $imagesize[0] holds the image width and $imagesize[1] holds the image height. Since we want the text to be in the bottom left corner, we set a slight x (horizontal) offset of 7px and a larger y (vertical) offset of the 20px less than the image height.

Overlay the Watermark

We need to allocate a color for our text watermark. In the code below, we are using white, but you can use any color you want. In traditional CSS, we use hex values for text colors (i.e. #FFFFFF). Each 2 digits corresponds to a red, green, or blue value. The PHP function takes these values as 3 separate arguments, thus we have *0xFF, 0xFF, 0xFF* passed in. If you wanted a color of #FF0034, you would use *0xFF, 0x00, 0x34* as the arguments in the function.

```
//allocate text color
$textcolor = imagecolorallocate($im, 0xFF, 0xFF, 0xFF);
```

The next bit of code is what actually overlays the text over our image. The first argument is our image resource ($im), then the font, x offset, y offset, watermark text, and the text color.

```
 //write out the watermark
imagestring($im, 5, $x_offset, $y_offset, 'watermark text goes here', $textcolor);
```

The watermark text can be anything you want (i.e. queried from a database, read from another file, passed in as a $_GET variable, etc).

Output the Image

Our final step is to output our watermarked image to the browser. These lines of code essentially tell the web browser to interpret our php script as a jpeg image.

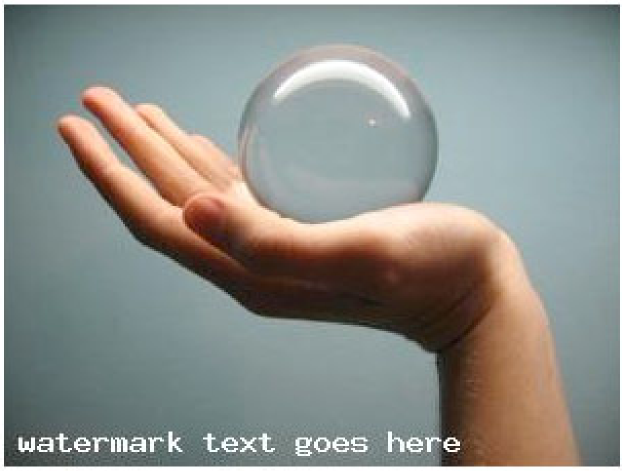
```
//output watermarked image
header('Content-type: image/jpg');
imagejpeg($im);
```

Test the Script

You can test out the script by uploading it to your local (http://localhost/image.php) or remote web server and just navigating to the URL.

Here is the stock example image:

And after running our PHP script with the watermark applied:



Roadmap Series

Issue 9

WebAssist.com Corporation
8899 University Center Lane
San Diego, CA 92122
USA
www.webassist.com

Final PHP Script

Your entire PHP script should look like the following:

```
<?php
//load image
$im = imagecreatefromjpeg('example.jpg');

//figure out where to put the text
$imagesize = getimagesize('example.jpg');
$x_offset = 7;
$y_offset = $imagesize[1] - 20;

//allocate text color
$textcolor = imagecolorallocate($im, 0xFF, 0xFF, 0xFF);

//write out the watermark
imagestring($im, 5, $x_offset, $y_offset, 'watermark text goes here', $textcolor);

//output watermarked image
header('Content-type: image/jpg');
imagejpeg($im);

?>
```

In Conclusion

This is just one example of using the GD library in PHP to create dynamic images. As you can see, the GD functions bundled in PHP are very powerful and can help automate image tasks for your web projects.

About WebAssist

WebAssist helps you build better websites faster by offering software, solutions, and training needed to succeed on the web. WebAssist.com hosts a thriving community of over 300,000 designers, developers, and business owners. WebAssist's partners include Adobe, Microsoft, and PayPal.

Join our Community

From free software and training, to special offers, there is something for everyone. Come take a look at what is waiting for you and sign up today to gain access to all of the valuable benefits awarded to the members of our community. Click here to join our community.