**Handling Images**

Besides its normal HTML duties, PHP also has functions to handle lots of different types of image files. The php\_gd2 extension library includes functions for creating and manipulating image files in various formats, such as GIF, JPEG, and PNG. You'll use some of the PHP image functions to help create and manipulate images for your storefront Web site. Before you can do that, though, you need to make sure you've installed the php\_gd2 extension in your AMP server.

**Installing the GD2 Extension**

The WampServer provides an excellent user interface for handling PHP extensions. All you need to do is find the php\_gd2 entry in the extensions menu and select it. Here are the steps to do that:

1. Start the WampServer services on your PC.
2. Click the WampServer icon in the system tray.
3. Select the **PHP** entry in the pop-up menu.
4. Select the **PHP Extensions** entry in the PHP Settings menu.
5. Select the **php\_gd2** entry in the PHP Extensions menu.
6. Select the **Stop All Services** entry from the WampServer menu.
7. Select the **Start All Services** entry from the WampServer menu.

You're now ready to start playing with images in your PHP applications.

|  |
| --- |
| **Note:** If you're using an ISP AMP server, you'll have to consult with your ISP to find out whether they have the php\_gd2 extension loaded. Depending on their setup, you may not be able to load it yourself. But it's a fairly common extension; so chances are they already have it loaded. |

**Creating a New Image File**

One of the exciting features in the GD2 library is the ability to create our own image files. Much like the Microsoft Paint program, the PHP GD2 library allows us to create a blank canvas and then draw lines, shapes, add text, set colors, and save it as an image file.

To create a new image, use the *imagecreatetruecolor()*function. This function takes two parameters: the width and height of the new image specified in pixels. It returns a value that you must use to reference the new image while working in the main program. For example, the code below creates a new image that is 80 pixels wide by 60 pixels high. This is a common image size for what's commonly called a *thumbnail* image. A thumbnail image is a smaller image that easily fits on a Web page.

$image = imagecreatetruecolor(80, 60);

After creating a new image, you'll probably want to draw in it. First, you must allocate colors to use for drawing your objects. The *imagecolorallocate()* function does this.

$bc = imagecolorallocate($image, 255, 255, 255);  
$fc = imagecolorallocate($image, 0, 0, 0, 0);

The imagecolorallocate() function takes four parameters. The first parameter is the image returned by the imagecreatetruecolor() function. The next three parameters are a color hue defined using the Red-Green-Blue (RGB) values commonly used in CSS style sheets. Each value is one byte in size, and you can specify them using either decimal or hexadecimal values. The combination 255, 255, 255 represents white, while the combination 0, 0, 0 represents black.

After allocating the colors you want to use, you can start drawing. The GD2 library includes several functions for drawing lines, shapes, and text.

|  |  |
| --- | --- |
| **Some GD2 Library Drawing Functions** | |
| **Function** | **Description** |
| imageline() | Draw a line between two specified points using a defined color. |
| imagechar() | Draw an alphanumeric character using a specified font, color, and location. |
| imagerectangle() | Draw a rectangle between four specified points using a specified color. |
| imagefilledrectangle() | Draw a solid rectangle between four specified points, filling it with a specified color. |
| imagestring() | Draw a string of characters using a specified font, color, and location. |

Let's build a small PHP program that creates a sample image file. In our storefront application, we'll need a generic image to display when an image for an item isn't available. Follow these steps to create that image.

* 1. Create a file in the WampServer www folder called *createimage.php*.
  2. Open the file using Notepad, and enter the following code:

<?php

$image = imagecreatetruecolor(80, 60);

$bc = imagecolorallocate($image, 255, 255, 255);  
$fc = imagecolorallocate($image, 0, 0, 0);

imagefilledrectangle($image, 0, 0, 80, 60, $bc);  
imagestring($image, 5, 20, 5, "No", $fc);  
imagestring($image, 5, 10, 20, "Image", $fc);  
imagestring($image, 5, 0, 35, "Available", $fc);

imagejpeg($image, "noimage.jpg");  
imagedestroy($image);

echo "Image created";  
?>

* 1. Save the file, and exit Notepad.
  2. Open a browser window, and go to http://localhost/createimage.php.

You should recognize most of the image functions here. The imagestring() function specifies a font size, followed by the X and Y coordinates of where you want the string to start, then the actual string, and finally, the color to use to draw the string.

The *imagejpeg()* function takes your newly created image and either displays it on the Web page or, in our case, writes it to a file as a JPEG image. After we're done, we use the *imagedestroy()* function to free up the memory associated with our image.

When you run the program, you won't see anything too exciting in your browser window—just a message saying the image was created. However, if you go to the www folder in the WampServer area, you'll see a new file called noimage.jpg! If you open that file with your default JPEG viewer, you should see your new image.

The noimage.jpg file

The imagestring() functions drew our strings on the image canvas to create the "No Image Available" image for our application. Set this image aside for now. We'll use it later.

**Modifying Existing Images**

One of the biggest problems in using images on Web pages is they are often too big. If you run a Web site that allows visitors to upload pictures, you never know quite what to expect. Some visitors upload picture files that are 300 x 500, while others upload picture files that are 1024 x 768! The trick is to standardize all of your images so they will fit nicely in your Web pages. Fortunately, the PHP GD library has just the tools for us!

The *imagecopyresampled()* function allows us to *resample*an existing image file to a new image. Resampling rebuilds the picture pixel by pixel at a different resolution using special algorithms to maintain the picture clarity. By resampling, we can either shrink or expand the image. The GD2 library handles all of the complicated mathematical routines required to do that. We'll use this technique to make all the images in our storefront Web site the same size.

In the meantime, here's some sample code you can use to create a test file and resample some JPEG images you have laying around:

<?php

$image = file\_get\_contents("test.jpg");  
$source = imagecreatefromstring($image);

$width = imagesx($source);  
$height = imagesy($source);

$thumb = imagecreatetruecolor(80, 60);  
imagecopyresampled($thumb, $source, 0, 0, 0, 0, 80, 60, $width, $height);  
imagejpeg($thumb, "newthumb.jpg");

echo "image created";  
?>

This example takes an original image called *test.jpg* and resizes it to a small thumbnail-sized image called *newthumb.jpg*.

First, we use the PHP function file\_get\_contents() to read the image file into a PHP variable. (Yes, you can store the entire contents of a file in a single PHP variable!) Next, we use the *imagecreatefromstring()* function that converts the contents of the variable into a new image we can manipulate.

In order to resize the image, the GD2 library needs to know the existing size of the image. We use the *imagesx()*function to get the width of the image and the *imagesy()*function to get the height of the image.

The next step is to create our new image that's the size we want using the imagecreatetruecolor() function. For this test, we'll use our 80 x 60 thumbnail size. Now that we have our original image and our new image, we use the *imagecopyresampled()* function to copy the original image to the new image using the new image size.

You must specify the destination image, source image, starting X and Y locations in the destination and source, and the width and height of the destination and source. To make our thumbnail image, we specify the destination width and height as 80 x 60.

Finally, we use the *imagejpeg()* function to convert the image to JPEG format and store it in a file called newthumb.jpg.

Let's test this out.

* 1. Create a new file in the WampServer folder called *makethumb.php*, and then copy the example code shown above into it. Save the file.
  2. Copy a JPEG image you have laying around to the WampServer www folder, and rename it *test.jpg*.
  3. Go to http://localhost/makethumb.php.

Again, the Web page won't be too exciting, but now look in your www folder. You should see the newthumb.jpg file. It will be a shrunken version of the original JPEG file you used for testing!

That's enough PHP code for one day. Follow me to the summary, and we'll wrap up today's lesson.