# 第22章 创 建 图 像

使用PHP能够完成的一件有意义的事情之一就是创建动态图像。PHP提供了一些内置的图像信息函数，也可以使用GD2函数库创建新图像或处理已有的图像。本章将介绍如何使用这些图像函数得到一些有趣而又有用的效果。

在本章中，我们将主要介绍以下内容：

- 在PHP中设定图像支持
- 理解图像格式
- 创建图像
- 在其他页面中使用自动生成的图像
- 用文本和字体创建图像
- 绘制图像与用图表描绘数据

特别地，我们将介绍两个例子：创建网站动态按钮和使用来自MySQL数据库中的数字绘制一个条形图。

在这里，我们将使用GD2函数库，但是还有一个颇受欢迎的PHP图像库。ImageMagick库并不是标准PHP的一部分，但是通过PHP扩展类库（PECL）很容易安装这个函数库。ImageMagick和GD2都有许多类似的特性，但是在某些方面，ImageMagick的功能更丰富一些。如果希望创建GIF（甚至是动画GIF），就应该使用ImageMagick。如果希望使用真彩图像或制造透明效果，应该比较这两个函数库所提供的功能。

访问如下站点可以找到ImageMagick的PHP下载扩展类库——PECL：

http: //pecl.php.net/package/imagick。

关于ImageMagick主要功能的介绍以及详细文档，请参阅如下站点：

http: //www.imagemagick.org。

## 22.1 在PHP中设置图像支持

在PHP中，有些图像函数是可以直接使用的，但是大多数函数需要安装GD2函数库。关于GD2的详细信息，请访问站点：http: //www.boutell.com/gd/。

从PHP的4.3版本开始，PHP捆绑了自己版本的GD2库，这是由PHP开发团队实现的。这个版本的GD2库更容易安装，因此我们可以使用这个版本。在Windows平台下，只要注册了php_gd2.dll 扩展，PNG和JPEG是自动支持的。注册php_gd2.dll非常简单，只要在PHP的安装目录（\ext子目录）找到该文件并复制到系统目录（如果使用Windows XP，就是C:\Windows\system）。此外还需要在php.ini文件中取消如下一行指令的注释（删除该行指令开始处的“;”），如下所示：

```
extension=php_gd2.dll
```

如果使用UNIX而又希望使用PNG，必须安装libpng库和zlib库，可以从如下站点分别获得它们：http: //www.libpng.org/pub/png/、http: //www.gzip.org/zlib/。

需要使用如下命令行选项对PHP进行配置：

```
--with-png-dir=/path/to/libpng
 -with-zlib-dir=/path/to/zlib
```

如果使用UNIX并且希望使用JPEG，必须下载jpeg-6b库，然后重新编译GD库，使其包括对JPEG的支持。可以从以下站点下载该库：ftp: //ftp.uu.net/graphics/jpeg/。

此外，还应该使用如下命令行选项重新配置和编译PHP：

```
--with-jpeg-dir=/path/to/jpeg-6b
```

如果希望在图像中使用TrueType字体，还需要FreeType库。这个函数库也是在从PHP 4版本开始捆绑的。当然，也可以从以下站点下载该库：http: //www.freetype.org/。

如果希望使用PostScript Type 1字体，必须下载t1lib库，可以从如下站点下载该库：

ftp: //sunsite.unc.edu/pub/Linux/libs/graphics/。

需要使用如下所示的命令行选项运行PHP的配置程序：

```
--with-t1lib[=path/to/t1lib]
```

最后使用--with--gd配置PHP。

## 22.2 理解图像格式

GD库支持JPEG、PNG和WBMP格式。但它不再支持GIF格式。下面，让我们简要地介绍一下这些格式。

### 22.2.1 JPEG

JPEG（发音为“jay-peg”）是联合图像专家组（Joint Photographic Experts Group，JPEG）的缩写，它是一种压缩标准的名字，我们提到JPEG文件的时候，其文件格式实际上是JFIF，它对应于JPEG组织发布的标准之一。

简单地说，JPEG通常是用来存储照片或者存储具有丰富色彩和色彩层次的图像。这种格式使用了有损压缩，也就是说，为了将图形压缩成更小的文件，图像质量有所破坏。因为JPEG压缩后应该包含基本类似的图像和颜色的层次，所以人眼可以忍受这些图像质量的损失。正是由于这个原因，该格式不适合绘制线条、文本或颜色块。

可以从官方网站了解更多关于JPEG/JFIF的信息：http: //www.jpeg.org/。

### 22.2.2 PNG

PNG（发音为“ping”）是可移植的网络图像（Portable Network Graphics, PNG）的缩写。该文件格式可以看作是图形文件交换格式（*Graphics Interchange Format, GIF*）的替代品，其原因我们将在后续内容详细介绍。PNG网站将其描述为“一种强壮的图像格式，并且是无损压缩”。正因为它是无损压缩，所以该图像格式适合包含文本、直线和简单颜色块的图像，例如，

Web站点标题和按钮——所有可以用GIF图像来达到的效果。通常，一个相同图像的PNG压缩版本大小与GIF压缩版本的大小相当。PNG还提供了可变的透明度、微细修正和二维空间交错。但是，它不支持动画——要实现动画，必须采用其扩展格式MNG，该格式仍在开发之中。

无损压缩模式对于图解来说是非常不错的，但是通常并不适合保存大图像，因为它们通常很大。

可以从官方网站了解更多关于PNG的信息：http: //www.libpng.org/pub/png/。

### 22.2.3 WBMP

WBMP全称为*Wireless Bitmap*（无线位图）。它是专门为无线通信设备设计的文件格式，但是并没有得到广泛应用。

### 22.2.4 GIF

GIF是图形文件交换格式（Graphics Interchange Format）的缩写，它是无损压缩格式，广泛应用于网络，用来存储包含文本、直线和单块颜色的图像。

GIF格式使用了来自24位RGB颜色空间的256种不同颜色的调色板。它还支持动画，允许每一帧使用不同的256色调色板。颜色的限制使得GIF格式不适用于产生高画质以及需要扩展颜色的图像，但是它还是非常适合简单图像，例如具有特定颜色区域的图形或徽标。

GIF使用LZW无损数据压缩技术进行压缩，这样就减少了文件大小，而又不会降低可视质量。

## 22.3 创建图像

在PHP中，创建一个图像应该完成如下所示的4个基本步骤：

1）创建一个背景图像，以后的操作都将基于此背景图像。

2）在背景上绘制图形轮廓或输入文本。

3）输出最终图形。

4）清除所有资源。

下面，我们先来了解一个非常简单的创建图像脚本。该脚本如程序清单22-1所示。

**程序清单22-1　simplegraph.php——输出带标签“Sales”的简单直线图形**

```
<?php
// set up image
  $height = 200;
  $width = 200;
  $im = imagecreatetruecolor($width, $height);
  $white = imagecolorallocate ($im, 255, 255, 255);
  $blue = imagecolorallocate ($im, 0, 0, 64);

// draw on image
  imagefill($im, 0, 0, $blue);
  imageline($im, 0, 0, $width, $height, $white);
  imagestring($im, 4, 50, 150, 'Sales', $white);
```

```
// output image
  Header ( 'Content-type: image/png');
  imagepng ($im);

// clean up
  imagedestroy($im);
?>
```

运行以上脚本，其输出结果如图22-1所示。


图22-1 该脚本绘制黑色背景，并在上面添加了一条直线和一个文本标签

接下来，我们将一步一步地讲述该图像的创建过程。

### 22.3.1 创建一个背景图像

要在PHP中开始创建或修改一个图像，必须首先建立一个图像标识符。有两种方法可以实现图像标识符的创建。一种方法是创建一个空白的背景，这可以通过调用函数ImageCreateTrueColor()来实现，如以下脚本所示：

```
$im = imagecreatetruecolor($width, $height);
```

我们需要为这个函数传递两个参数。第一个是新图像的宽度，另一个是新图像的高度。该函数将返回新图像的标识符。(它非常类似于文件句柄。)

另一种方法是读入一个已有的图像文件，可以对图像进行过滤，改变其大小或在上面添加其他图像。根据所读入的文件格式不同，可以通过调用下面3个函数之一来实现：Imagecreatefrompng()、Imagecreatefromjpeg()或Imagecreatefromgif()。这3个函数都是以文件名作为参数的，如下所示：

```
$im = imagecreatefrompng( 'baseimage.png' );
```

本章后续内容所示的一个例子说明了如何使用已有的图像来创建动态按钮。

### 22.3.2 在图像上绘图或打印文本

在一个图像上绘图或打印文本需要两个步骤。首先，必须选择希望绘制的颜色。我们知道，在计算机显示器上显示的颜色是由不同数量的红色、绿色和蓝色组成的。图像格式使用一个调色板，它包含所有3种颜色的可能组合的特定子集。要使用一种颜色绘制一个图像，必须将此颜色添加到图像的调色板上。我们必须对每一种要使用的颜色进行相同的处理，即使它是白色或黑色。

也可以通过调用Imagecolorallocate()函数为图像选择颜色。需要为该函数传递图像标识符和希望使用的颜色，而颜色由红、绿和蓝（RGB）值的组合决定。

在程序清单22-1中，我们使用了两种颜色：黑色和白色。通过调用以下函数分配这两种颜色。

```
$white = imagecolorallocate ($im, 255, 255, 255);
$blue = imagecolorallocate ($im, 0, 0, 64);
```

以上函数将返回一个可供以后使用的颜色标识符。

其次，要真正将颜色绘制到图像中，还需要使用许多其他不同的函数，对这些函数的使用取决于要绘制的内容——直线、弧、多边形或文本。

通常，绘图函数需要下列参数：

- 图像标识符
- 需要绘制内容的起始坐标和结束坐标
- 图像使用的颜色
- 对于文本，需要字体信息

在这个例子中，我们使用了3个绘画函数。下面，我们逐一了解这些函数。

首先，调用ImageFill()函数绘制了一个用以在上面绘画的黑色背景：

```
imagefill($im, 0, 0, $blue);
```

该函数以图像标识符、绘画区域的起始坐标（$x$和$y$）以及颜色作为参数。

**提示**　需要注意的一点是，图像的起始坐标从左上角开始，该点坐标为$x$=0、$y$=0。图像右下角的坐标是$x$ = $width、$y$ = $height。这是计算机图形的常识，但是这与常规的作图习惯是相反的。

接下来，我们从左上角（0，0）开始画一条线，直到图像的右下角（$width ,$height）：

```
imageline($im, 0, 0, $width, $height, $white);
```

该函数以图像标识符、直线的起始点的$x$和$y$坐标、终点以及颜色作参数。

最后，我们在该图中添加了一个标签：

```
imagestring($im, 4, 50, 150, 'Sales , $white);
```

Imagestring()函数所需的参数与Imageline()有些不同。其原型是：

```
int imagestring (resource im, int font, int x, int y, string s, int col)
```

它以图像标识符、字体、文本的起始坐标$x$与$y$以及颜色作为参数。

字体参数值是1～5的数字。它们表示一组以latin2为编码的内置字体，参数值越高，对应的字体越大。也可以选择特殊字体，例如，可以选择TrueType字体，或PostScript Type 1字体。这些字体都有相应的函数设置。在下一个例子中，我们将使用TrueType字体的函数。

使用可选的字体函数组的一个原因就是由函数Imagestring()或相关函数如Imagechar()（写一个字符到图像）写出的文本是可用别名的，而TrueType和PostScript函数生成反别名的文本。


图22-2　常规的文本表现出交错的锯齿，特别是字体很大的时候，而反别名

如果不能确认它们到底有什么区别，请参

阅图22-2。请注意，字母中的曲线或折线的地方，别名文本将表现出交错的锯齿。而曲折和拐弯的地方则是通过使用“staircase”效果而获得的。在反别名的图像中，文本中曲线或折线的地方以及背景色和文本色之间的颜色趋于平滑。

### 22.3.3 输出最终图形

可以将一个图像直接输出到浏览器或者一个文件。

在这个例子中，我们将图像直接输出到浏览器。这包括两个步骤。首先，需要告诉Web浏览器我们输出的是一个图像而不是文本或HTML。这可以通过调用Header()函数指定图像的MIME类型来完成：

```
Header ( "Content-type: image/png " );
```

通常，当在浏览器中接收一个文件时，Web服务器首先发送的内容是MIME类型。对于一个HTML或PHP页面（执行后），最先发送的是：

```
Content-type: text/html
```

它将告诉浏览器如何解释后续的数据。

在这个例子中，我们要告诉浏览器将发送的是一个图像而不是常规的HTML输出。可以调用函数Header()来实现它。该函数我们尚未讨论过。

该函数将发送一个HTML标题字符串。该函数的另一个典型应用是实现HTTP重定向，告诉浏览器来加载一个不同的页面，而不是被请求的那个页面。它们通常应用于删除页面的时候。例如：

```
Header ( "Location: http://www.domain.com/new_home_page.html ");
```

值得注意的是，在使用Header()函数时，如果该页的HTTP标题已经被发送了，那么该函数将不会执行。一旦输出任何东西到浏览器，PHP将自动发送一个HTTP标题。因此，如果使用了任何echo语句，或者甚至是在打开PHP标记之前有任何空白区域，该标题都将被发送过去，并且在试图调用Header()的时候，会从PHP接收到一个警告信息。然而，你却可以在同一个脚本里多次调用Header()函数，从而发送多个HTTP标题，尽管它们都必须出现在向浏览器发送任何输出之前。

在我们发送标题数据之后，将通过调用如下函数输出图像数据：

```
imagepng ($im);
```

该函数以PNG格式将输出内容发送到浏览器。如果希望以不同的格式发送，可以调用Imagejpeg()函数（如果系统支持JPEG的话）。仍需首先发送相应的标题，也就是：

```
Header ( "Content-type: image/jpeg " );
```

当然，也有第二选择，那就是：将图像发送到一个文件而不是浏览器。这可以通过将可选的第二个参数加到ImagePNG()函数（或者，与之类似的支持其他格式的函数）来实现：

```
imagepng($im, $filename);
```

请记住，在这里，所有PHP的写文件规则都适用（例如，正确地设置文件权限）。

### 22.3.4 清理

当完成对一个图像的处理的时候，应该通过销毁图像标识符将所占用的所有资源返回给服务器。可以通过调用Imagedestroy()函数来完成：

```
imagedestroy();
imagedestroy($im);
```

## 22.4 在其他页面中使用自动生成的图像

因为标题只可以发送一次，而这是告诉浏览器正在发送图像数据的唯一方法，所以在普通页面里嵌入动态图像会遇到一些麻烦。可以通过以下3种方法实现：

1. 正如我们在前面的例子中，可以拥有一个由图像输出组成的页面。

2. 正如前面所提到的，可以将图像写到一个文件中，然后用<IMG>标记指向它。

3. 可以将图像创建脚本置于一个图像标记中。

我们已经介绍过前两种方法。下面简要介绍一下第三种方法。要采用这种方法，需要通过图像标记包含一个HTML内嵌图像，如下所示：

```
<img src='simplegraph.php'height= '200"
width= "200'alt="Sales going down"/>
```

除了直接将PNG、JPEG或GIF图像加入到IMG标记，还可以在SRC属性中使用能够生成图像的PHP脚本。该图像可以被检索，其输出是内嵌式的，如图22-3所示。


图22-3　动态产生的内嵌图像对终端用户来说看起来像普通的图像

## 22.5 使用文本和字体创建图像

我们来看一个关于创建图像更复杂的例子。如果能够在网站里自动创建一个按钮或其他图像，那将会是很有趣的。可以使用我们已经介绍的技术在基于一个四边形的背景颜色上创建一个简单的按钮。通过编程，还可以实现更加复杂的效果，但是通常使用一个画板程序更容易完成这些操作。这也使得我们让一个艺术家来绘制图形而程序员来完成编程工作。

在这个例子中，我们将使用一个空白按钮模板来创建多个按钮，这样，通过该按钮模板创建的按钮具有相同的特征（如羽化边缘等），这些特征在使用Photoshop、GIMP或其他图形工具创建时容易得多。使用PHP中的图像库，我们可以从一个基本图像开始，在此图像上绘制。

我们也使用TrueType字体以便可以使用反别名的文本。TrueType字体函数有它们自己的特性，关于这些内容，我们将在以后讨论。

基本的处理过程是接收一些文本并产生包含此文本的按钮，该文本位于按钮的中央（水平方向和垂直方向都处于中央），并被赋以适合按钮的最大字体。

我们已经创建了用于测试和试验的按钮生成器的前台，其界面如图22-4所示。（因为这非

常简单，我们没有为表单包含进HTML，但是可以在本书附带的文件找到，在design_button.html文件中。）

可以使用这种界面让程序自动创建网站。也可以通过联机的方式调用该脚本来创建网站所有的动态按钮，但是这可能需要缓冲机制，防止其占用太多的处理器时间。

该脚本的输出结果如图22-5所示。

该按钮由make_button.php脚本生成。脚本如程序清单22-2所示。


图22-4 前台允许用户选择的请求文本中按钮的颜色和类型


图22-5 make_button.php脚本生成的按钮

**程序清单22-2 make_button.php——该脚本可以从design_button.html的表单或一个HTML图像标签中调用**

```
<?php
// check we have the appropriate variable data
// variables are button_text and color

$button_text = $_REQUEST['button_text'];
$color = $_REQUEST['color'];

if (empty($button_text) || empty($color))
{
  echo 'Could not create image - form not filled out correctly';
  exit;
}

// create an image of the right background and check size
$im = imagecreatefrompng ($color.'-button.png');

$width_image = imagesx($im);
$height_image = imagesy($im);

// Our images need an 18 pixel margin in from the edge of the image
$width_image_wo_margins = $width_image - (2 * 18);
$height_image_wo_margins = $height_image - (2 * 18);
```

```
// Work out if the font size will fit and make it smaller until it does
// Start out with the biggest size that will reasonably fit on our buttons
$font_size = 33;

// you need to tell GD2 where your fonts reside
putenv('GDFONTPATH=C:\WINDOWS\Fonts');
$fontname = 'arial';

do
{
  $font_size--;

  // find out the size of the text at that font size
  $bbox=imagettfbbox ($font_size, 0, $fontname, $button_text);

  $right_text = $bbox[2]; // right co-ordinate
  $left_text = $bbox[0];  // left co-ordinate
  $width_text = $right_text - $left_text;  // how wide is it?
  $height_text = abs($bbox[7] - $bbox[1]); // how tall is it?
  }
  while ( $font_size>8 &&
          ( $height_text>$height_image_wo_margins ||
            $width_text>$width_image_wo_margins )
        );

if ( $height_text>$height_image_wo_margins ||
     $width_text>$width_image_wo_margins )
{
  // no readable font size will fit on button
  echo 'Text given will not fit on button.<br />';
}
else
{
  // We have found a font size that will fit
  // Now work out where to put it

  $text_x = $width_image/2.0 - $width_text/2.0;
  $text_y = $height_image/2.0 - $height_text/2.0 ;

  if ($left_text < 0)
      $text_x += abs($left_text); // add factor for left overhang

$above_line_text = abs($bbox[7]); // how far above the baseline?
$text_y += $above_line_text;      // add baseline factor

$text_y -= 2; // adjustment factor for shape of our template
```

```
    $white = imagecolorallocate ($im, 255, 255, 255);

    imagettftext ($im, $font_size, 0, $text_x, $text_y, $white, $fontname,
                  $button_text);

    Header ('Content-type: image/png');
    imagepng ($im);
}

imagedestroy ($im);
?>
```

这是我们所看到的最长的脚本了。我们将逐步地讨论脚本的各个部分。开始是一些基本的错误检查，然后是创建画布并在此画布上操作。

### 22.5.1 创建基本画布

在程序清单22-2中，我们并不是从零开始作图，而是以一个已有的按钮图像为开始的。基本按钮的颜色存在3种选择：红（red-button.png）、绿（green-button.png）和蓝(blue-button.png)。

用户选择的颜色将保存在来自表单的变量color中。首先我们将从超级全局变量$_REQUEST中获得颜色，并且创建一个基于适当按钮的新图像标识符：

```
$color = $_REQUEST['color1'];
...
$im = imagecreatefrompng ($color.'-button.png');
```

函数Imagecreatefrompng()以一个PNG文件名作为参数，并且返回一个包含该PNG图像拷贝的图像标识符。请注意，这里并没有对基本的PNG进行任何修改。如果已经安装了适当的支持程序，就可以使用Imagecreatefromjpeg()和ImageCreateFromGIF()函数。

**提示** 对Imagecreatefrompng()函数的调用只是在内存中创建图像。要将该图像保存到一个文件或输出到浏览器，必须调用ImagePNG()函数。后面我们将讨论它，此前我们需要先对图像进行一些其他处理。

### 22.5.2 将文本调整到适合按钮

我们已经接收到了用户键入的一些文本，这些文本保存在$button_text变量中，我们要做的是将它以适合按钮的最大字体打印在按钮上。通过多次调整可以达到此目标；严格地说，是经过反复的试验。

首先设置一些相关变量。首要的两个变量是按钮图像的高度和宽度：

```
$width_image = imagesx($im);
$height_image = imagesy($im);
```

其次，设置两个表示按钮边缘的宽度的变量。按钮图像是有斜边的，因此应在文本周围为

斜边留有空白。如果使用的图像与此不同，这个数字将是不同的！在这个例子中，每边的空白大概是18个像素：

```
$width_image_wo_margins = $width_image - (2 * 18);
$height_image_wo_margins = $height_image   (2 * 18);
```

我们还设置了初始化字体大小。起始字体大小是32（实际上是33，但是我们会马上减小它），因为它大约是可以适合该按钮的最大字体：

```
$font_size = 33;
```

使用GD2库，必须通过设置环境变量GDFONTPATH告诉脚本字体所在的位置，如下所示：

```
putenv( 'GDFONTPATH=C:\WINDOWS\Fonts' );
```

我们还需要设置希望使用的字体名称。我们将在TrueType函数中使用这个字体，这将在以上字体路径中查找字体文件，而且将在文件名称后添加.ttf扩展名（TrueType字体）。

```
$fontname = 'arial';
```

请注意，根据操作系统的不同，可能要在字体名称后添加“.ttf”。

如果系统没有安装Arial（我们在这个例子中使用的字体）字体，可以将其改变为其他的TrueType字体。

现在我们开始执行循环，每次字体大小递减1，如此重复，直到提交的文本基本适合按钮：

```
do
{
  $font_size--;

  // find out the size of the text at that font size
  $bbox=imagettfbbox ($font_size, 0, $fontname, $button_text);

  $right_text = $bbox[2]; // right co-ordinate
  $left_text = $bbox[0];  // left co-ordinate
  $width_text = $right_text - $left_text;  // how wide is it?
  $height_text = abs($bbox[7] - $bbox[1]); // how tall is it?

}
while ( $font_size>8 &&
        ( $height_text>$height_image_wo_margins ||
          $width_text>$width_image_wo_margins )
);
```

以上代码通过查看所谓的文本边框测试文本的大小。通过调用ImagegetttfBox()函数完成测试，该函数是TrueType字体函数之一。在计算出字体大小之后，我们将用TrueType字体（在这个例子中，我们使用的是Arial字体，但也可以使用任何你所喜欢的字体），以及调用Imagettftext()函数将文本打印在按钮上。


图22-6　边框的坐标是相对于基线给出的。坐标原点是图中的（0, 0）

一段文本的边框是围绕该文本的最小可能边框。边框的一个示例如图22-6所示。

要获取该框的大小，可以调用如下函数：

```
$bbox=imagettfbbox ($font_size, 0, $fontname, $button_text);
```

以上调用的意思是，“对给定的字体大小$font_size，其文本倾斜0°，它使用TrueType字体Arial，请告诉我$button_text变量中文本的大小。”

请注意，实际上还需要将包含该字体文件的路径传递给该函数。在这个例子中，其目录与脚本所在目录相同（在默认情况下），因此不用指定其他更长的路径。

该函数将返回一个包含边框各个角的坐标的数组。数组内容如表22-1所示。

表22-1 边框数组的内容

| 数组索引 | 内 容 | 数组索引 | 内 容 |
|---|---|---|---|
| 0 | x坐标，左下角 | 4 | x坐标，右上角 |
| 1 | y坐标，左下角 | 5 | y坐标，右上角 |
| 2 | x坐标，右下角 | 6 | x坐标，左上角 |
| 3 | y坐标，右下角 | 7 | y坐标，左上角 |

要记住数组的内容，只要记住坐标开始于边框的左下角，并且坐标系是逆时针方向的。

应当注意的是，从ImageTTFBBox()函数返回的值。这些值都是坐标值，也就是与原点的距离的相对值。然而，它们不像图像坐标，图像坐标相对于图像左上角而定，而它们是根据基线而定的。

让我们回头再看看图22-6。可以看到，我们沿大部文本的底部画了一条线。该线称为基线。一些字母在行方向上低于基线，例如，在这个例子中的字母“y”。我们称之为下行字母。

基线的左边定义为起始量度——也就是，x坐标为0，y坐标也为0。坐标位于x之上称x坐标为正，位于x之下称x坐标为负。

除此之外，事实上，文本的坐标值还有位于边框之外的可能情况。例如，文本可能实际上从x坐标值-1的地方开始。

这些事实都使我们用这些数字进行计算的时候要加倍小心。

我们将按如下步骤计算文本的宽度和高度：

```
$right_text = $bbox[2]; // right co-ordinate
$left_text = $bbox[0];  // left co-ordinate
$width_text = $right_text - $left_text;  // how wide is it?
$height_text = abs($bbox[7] - $bbox[1]); // how tall is it?
```

完成这些操作后，我们将测试循环条件。

```
} while ( $font_size>8 &&
          ( $height_text>$height_image_wo_margins |
            $width_text>$width_image_wo_margins )
          );
```

在这里，我们测试了两组条件。首先是字体是否仍然可读——使用小于8号的字体已经没有意义，因为人的肉眼几乎不能看清按钮上的文本。第二组条件是测试文本是否适合我们为其

绘制的空间。

下一步，将检查循环计算是否找到了可以接受的字体大小。如果没有找到，将报告错误：

```
if ( $height_text>$height_image_wo_margins ||
     $width_text>$width_image_wo_margins )
{
  // no readable font size will fit on button
  echo "Text given will not fit on button.<br />";
}
```

### 22.5.3 放置文本

如果前面所有的工作都做好了，下面就可以开始计算文本的基本位置。计算的是有效区域的中点。

```
$text_x = $width_image/2.0 - $width_text/2.0;
$text_y = $height_image/2.0 - $height_text/2.0 ;
```

因为使用相对基线的坐标系统比较复杂，我们需要添加一些矫正因子：

```
if ($left_text < 0)
    $text_x += abs($left_text);      // add factor for left overhang
$above_line_text = abs($bbox[7]);    // how far above the baseline?
$text_y += $above_line_text;         // add baseline factor
$text_y -= 2;     // adjustment factor for shape of our template
```

因为我们的图形太复杂，矫正因子允许采用基线和微调。

### 22.5.4 将文本写到按钮上

此后的事情就非常简单了。将文本颜色设置为白色：

```
$white = ImageColorAllocate ($im, 255, 255, 255);
```

然后，可以调用ImageTTFText()函数将文本写到按钮上：

```
imagettftext ($im, $font_size, 0, $text_x, $text_y, $white, $fontname,
                $button_text);
```

该函数需要许多输入参数。依次是：图像标识符、用像素表示的字体大小、文本倾斜角度、文本的起始*x*和*y*坐标、文本颜色字体，以及写到按钮的实际文本。

**提示**　字体文件需要保存在服务器上可供读取的位置，它不能从客户端机器上获得，因为客户端的机器将它看作一个图像。

### 22.5.5 完成

最后，可以将该按钮输出到浏览器：

```
Header ( "Content-type: image/png");
```

```
imagepng ($im);
```

然后，要清除占用的资源并且结束脚本：

```
imagedestroy ($im);
```

这就完成了！如果一切顺利，应当看到已经有一个按钮显示在浏览器中了，它看起来如图22-5所示。

## 22.6 绘制图像与用图表描绘数据

在上一个应用程序中，我们已经使用了已有图像和文本。我们还没有了解绘图的例子，现在就介绍它。

在这个例子中，我们将在本网站中进行一次民意测验，让用户为虚构的选举进行投票。投票结果保存到一个MySQL数据库中，并用图像函数绘制出条形图，以表示投票结果。

这些函数的另一个主要应用是绘制图形。还可以用图形来表示任何需要表示的数据——交易量、网站点击数和任何我们所喜欢的。

对于这个例子，我们花了几分钟建立了一个名为poll的MySQL数据库。该数据库包含一个名为poll_results的表，该表的candidate列保存了候选人的名字，num_votes列保存了候选人收到的投票数。我们还为该数据库创建了一个名为poll的用户，其密码为poll。这个表的创建过程简单明了，可以通过运行如程序清单22-3所示的SQL脚本完成。可以使用如下命令，以root用户的身份进行登录：

```
mysql -u root -p < pollsetup.sql
```

当然，也可以以具有适当权限的用户身份进行登录。

**程序清单22-3 pollsetup.sql——创建Poll数据库**

```
create database poll;
use poll;
create table poll_results (
  candidate varchar(30),
  num_votes int
);
insert into poll_results values
  ('John Smith', 0),
  ('Mary Jones', 0),
  ('Fred Bloggs', 0)
;
grant all privileges
on poll.*
to poll@localhost
identified by 'poll';
```

该数据库包含了三个候选人。我们通过vote.html页面提供一个投票界面。该页面代码如程序清单22-4所示。

**程序清单22-4　vote.html——用户可以在此投票**

```
<html>
<head>
  <title>Polling</title>
<head>
<body>
<h1>Pop Poll</h1>
<p>Who will you vote for in the election?</p>
<form method= "post"action= "show_poll.php ">
<input type= "radio"name= 'vote "value= 'John Smith'>John Smith<br />
<input type= "radio"name= "vote "value= 'Mary Jones'>Mary Jones<br />
<input type= "radio"name= "vote "value= 'Fred Bloggs">Fred Bloggs<br /><br />
<input type= "submit "value= 'Show results '>
</form>
</body>
</html>
```

该页面的输出结果如图22-7所示。

通常的想法是，当用户点击提交按钮时，将他们的投票添加到数据库中，然后再将数据库中所有的投票取出，绘制当前结果的条形图。

在用户投票之后的条形图输出如图22-8所示。

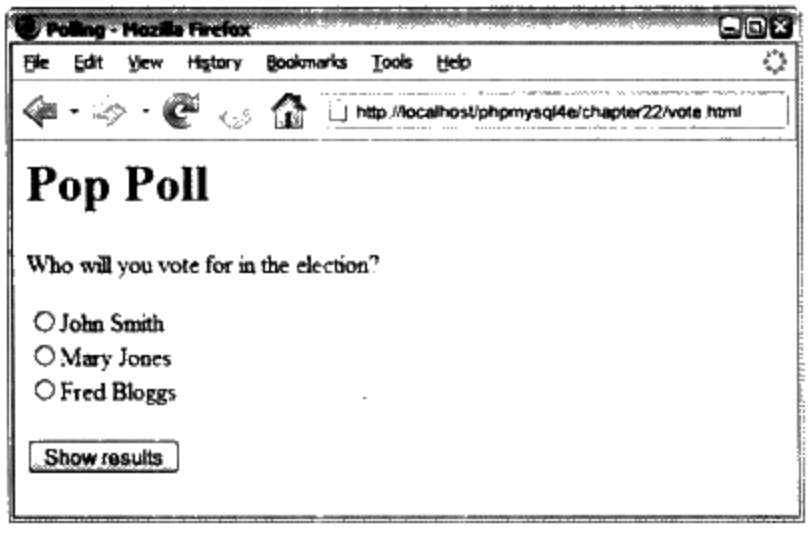


图22-7　用户可以在此投票，点击提交按钮将显示当前投票结果

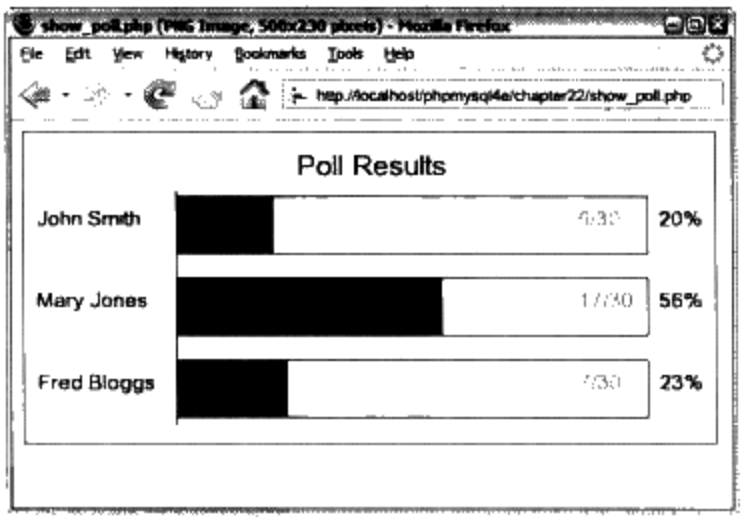


图22-8　投票结果通过将一系列线条、矩形和文本绘制到背景图来实现

创建该图像的脚本非常长。我们将它分为4部分，并分别加以讨论。这些脚本大多数都是我们非常熟悉的；我们已经看过许多类似的MySQL示例。我们已经了解了如何用单一的颜色绘制一个背景画布，以及如何在背景图上打印文本标签。

与前面所介绍的脚本相比较，该脚本新的部分是绘制线条和矩形。我们将重点讨论这些部分。第1部分（脚本1个部分的第1部分）如程序清单22-5-1所示。

第1部分，如程序清单22-5-1所示，将连接到MySQL数据库，根据用户的投票更新投票数，获得新票数。在获得新票数信息后，就可开始通过计算绘制图表。第2部分代码如程序清单22-5-2所示。

**程序清单22-5-1 show_poll.php——脚本的第1部分将更新投票数据库并检索新的投票结果**

```
<?php
/*********************************************
  Database query to get poll info
*********************************************/

// get vote from form
$vote=$_REQUEST['vote'];

// log in to database
if (!$db_conn = new mysqli('localhost', 'poll', 'poll', 'poll' ))
{
  echo 'Could not connect to db<br />';
  exit;
}

if (!empty($vote)) // if they filled the form out, add their vote
{
  $vote = addslashes($vote);
  $query = "update poll_results
            set num_votes = num_votes + 1
            where candidate = '$vote'";
  if(!($result = @$db_conn->query($query)))
  {
    echo 'Could not connect to db<br />';
    exit;
  }
}

// get current results of poll, regardless of whether they voted
$query = 'select * from poll_results';
if(!($result = @$db_conn->query($query)))
{
  echo 'Could not connect to db<br />';
  exit;
}
$num_candidates = $result->num_rows;
// calculate total number of votes so far
$total_votes=0;
while ($row = $result->fetch_object())
{
    $total_votes += $row->num_votes;
}
$result->data_seek(0); // reset result pointer
```

**程序清单22-5-2　show_poll.php——脚本的第2部分设置绘画所需要的所有变量**

```
/*********************************************
  Initial calculations for graph
*********************************************/
// set up constants
putenv('GDFONTPATH=C:\WINDOWS\Fonts');
$width=500;         // width of image in pixels - this will fit in 640x480
$left_margin = 50; // space to leave on left of graph
$right_margin= 50; // ditto right
$bar_height = 40;
$bar_spacing = $bar_height/2;
$font = 'arial';
$title_size= 16; // point
$main_size= 12;  // point
$small_size= 12; // point
$text_indent = 10; // position for text labels from edge of image

// set up initial point to draw from
$x = $left_margin + 60;  // place to draw baseline of the graph
$y = 50;                 // ditto
$bar_unit = ($width-($x+$right_margin)) / 100;     // one 'point' on the graph

// calculate height of graph - bars plus gaps plus some margin
$height = $num_candidates * ($bar_height + $bar_spacing) + 50;
```

脚本的第2部分创建了一些用来实际绘制图形的变量。

计算出这些类型的变量值是冗长乏味的，但是想想完成之后的图像是什么模样，会使这个过程变得轻松一点。在这里，我们使用的值是通过在纸上拟定设计效果，然后再估计所要求的比例而得到的。

宽度变量$width是背景总宽度。我们也设置了左边缘和右边缘的宽度（分别使用$left_margin和$right_margin变量），每条长方形的厚度和它们之间的间距（$bar_height和$bar_spacing），以及字体、字体大小和标签的位置（$font、$title_size、$main_size、$small_size和$text_indent）。

在给定这些基本的值后，就可以进行一些基本的计算了。我们希望绘制一条基线，所有的横条都从该线伸展开。通过在左边缘加上文本标签的宽度可以得到基线的*x*坐标位置，然后根据边框线估计基线的*y*坐标位置。如果灵活性非常重要，还可以根据最长的名称获得确切的宽度。

我们还需要计算两个重要的值：第一，在图像上表示一个单元的宽度：

```
$bar_unit = ($width-($x+$right_margin)) / 100;     // one 'point' on the graph
```

这是长方形条的最大宽度——从基线到右边空白处一分为100份，因为我们的图形要显示百分值。

第二个值是背景总高度：

```
$height = $num_candidates * ($bar_height + $bar_spacing) + 50;
```

这基本上是每一条长方形的高度乘以其数据，再加上标题所需的高度。脚本的第3部分如程序清单22-5-3所示。

**程序清单22-5-3 show_poll.php——脚本的第3部分建立图形并准备添加数据**

```
/*********************************************
  Set up base image
*********************************************/
// create a blank canvas
$im = imagecreatetruecolor($width,$height);

// Allocate colors
$white=imagecolorallocate($im,255,255,255);
$blue=imagecolorallocate($im,0,64,128);
$black=imagecolorallocate($im,0,0,0);
$pink = imagecolorallocate($im,255,78,243);

$text_color = $black;
$percent_color = $black;
$bg_color = $white;
$line_color = $black;
$bar_color = $blue;
$number_color = $pink;

// Create "canvas" to draw on
imagefilledrectangle($im,0,0,$width,$height,$bg_color);

// Draw outline around canvas
imagerectangle($im,0,0,$width-1,$height-1,$line_color);

// Add title
$title = 'Poll Results ';
$title_dimensions = imagettfbbox($title_size, 0, $font, $title);
$title_length = $title_dimensions[2] - $title_dimensions[0];
$title_height = abs($title_dimensions[7] - $title_dimensions[1]);
$title_above_line = abs($title_dimensions[7]);
$title_x = ($width-$title_length)/2; // center it in x
$title_y = ($y - $title_height)/2 + $title_above_line; // center in y gap
imagettftext($im, $title_size, 0, $title_x, $title_y,
             $text_color, $font, $title);

// Draw a base line from a little above first bar location
// to a little below last
imageline($im, $x, $y-5, $x, $height-15, $line_color);
```

在第3部分，我们将建立基本图像，分配颜色，然后开始绘制图形。

首先，我们将填充图形的背景颜色，使用如下所示函数：

```
imagefilledrectangle($im,0,0,$width,$height,$bg_color);
```

顾名思义，ImageFilledRectangle()函数将绘制一个中间填充了颜色的矩形。该函数的第一个参数与其他图形函数一样都是图像标识符。然后我们将此矩形的起点和结束点的$x$与$y$坐标传递给该函数。其起始点和结束点分别对应矩形的左上角和右下角。在这个例子中，我们还在整个背景中填充了背景颜色，该颜色是最后一个参数：白色。

然后调用：

```
imagerectangle($im,0,0,$width-1,$height-1,$line_color);
```

如上语句将绘制出围绕画布边沿的黑色轮廓线。该函数绘制了一个轮廓矩形而不是实心矩形。而其参数与前面的函数是一样的。请注意，我们绘制的矩形坐标是一直绘制到$width-1和$height-1——从（0，0）一直到画布的宽度和高度。如果我们绘制的是$width和$height，矩形将超出画布区域。

在这里，我们使用了前一个示例所介绍的逻辑和函数，将标题写到图像中，并将其置于图像中央。

最后，我们为每个长方形图绘制了基线：

```
imageline($im, $x, $y-5, $x, $height-15, $line_color);
```

ImageLine()函数可以在我们所指定的图像（$im）上画一条线，我们指定了该直线的起点坐标（$x，$y-5）和终点坐标（$x,$height-15），并通过变量$line_color指定颜色。

在这个例子中，我们在第一条长方形图的上面一点绘制了基线，其终点是背景的底端一点。

现在，我们准备将数据填充到该图形上。脚本的第4部分如程序清单22-5-4所示。

**程序清单22-5-4　show_poll.php——脚本的第4部分将实际的数据绘制到该图像上，并结束整个程序**

```
/*********************************************
  Draw data into graph
*********************************************/
// Get each line of db data and draw corresponding bars
while ($row = $result->fetch_object())
{
  if ($total_votes > 0)
    $percent = intval(($row->num_votes/$total_votes)*100);
  else
  $percent = 0;

  // display percent for this value
  $percent_dimensions = imagettfbbox($main_size, 0, $font, $percent.'%');
  $percent_length = $percent_dimensions[2] - $percent_dimensions[0];
  imagettftext($im, $main_size, 0, $width-$percent_length-$text_indent,
               $y+($bar_height/2), $percent_color, $font, $percent.'%');

  // length of bar for this value
  $bar_length = $x + ($percent * $bar_unit);

  // draw bar for this value
```

```
  imagefilledrectangle($im, $x, $y-2, $bar_length, $y+$bar_height, $bar_color);

  // draw title for this value
  imagettftext($im, $main_size, 0, $text_indent, $y+($bar_height/2),
               $text_color, $font, "$row->candidate");

  // draw outline showing 100%
  imagerectangle($im, $bar_length+1, $y-2,
                 ($x+(100*$bar_unit)), $y+$bar_height, $line_color);

  // display numbers
  imagettftext($im, $small_size, 0, $x+(100*$bar_unit)-50, $y+($bar_height/2),
               $number_color, $font, $row->num_votes.'/'.$total_votes);

  // move down to next bar
  $y=$y+($bar_height+$bar_spacing);
}
/*********************************************
  Display image
*********************************************/
Header( "Content-type: image/png");
imagepng($im);

/*********************************************
  Clean up
*********************************************/
imagedestroy($im);
?>
```

第4部分脚本将在数据库中逐个检索候选人，计算各自投票百分比，并为每位候选人绘制条形图并写上标签。

这里，我们又调用了ImageTTFText()函数来为图像添加标签。我们调用ImageFilledRectangle()函数绘制一个实心条形图：

```
imagefilledrectangle($im, $x, $y-2, $bar_length, $y+$bar_height, $bar_color);
```

并使用ImageRectangle()函数绘制了标志100%的轮廓：

```
imagerectangle($im, $bar_length-1, $y-2,
               ($x+(100*$bar_unit)), $y+$bar_height, $line_color);
```

在完成所有这些条形图之后，调用ImagePNG()函数输出了图像，并调用ImageDestroy()函数清理资源。

这是一段比较长的代码，但是它合乎需要，或者适合通过一个界面自动产生投票结果。该脚本一个重要的缺陷是它缺乏反欺骗机制。用户很快就可以发现他们可以重复投票而使投票结果变得毫无意义。

我们可以使用类似的方法绘制直线图形，甚至如果擅长数学的话，还可以用这样的方法来

绘制杂乱的流程图。

## 22.7　使用其他图像函数

除了本章中用到的图像函数之外，PHP还提供了许多其他的图像函数。通过编程来绘制图形将要花费很长时间，经过多次尝试和遇到很多错误。绘图的时候我们通常应该先绘制出其轮廓，然后查阅联机手册以寻找所需的函数。

## 22.8　进一步学习

许多可供阅读的资料可以通过在线方式获得。如果在使用这些图像函数时遇到了问题，查看GD库的源代码文档可能很有帮助，因为PHP函数只是封装了该库中的函数。关于GD库的文档，可以从以下站点获取：http://www.libgd.org/Documentation。

但是，请记住，GD2的PHP版本只是主函数库的一个派生库，因此许多实现细节是不同的。

还有许多关于特定类型的图形应用的优秀教程。尤其是在Zend和Devshed网站，其URL如下所示：http://www.zend.com and http: //devshed.com, respectively。

创建本章示例条形图应用程序的灵感也是来自Steve Maranda所写的动态图像脚本。该脚本可以在Devshed站点找到。

## 22.9　下一章

在下一章中，我们将讨论在PHP中使用方便的会话控制功能。