**Warning**: extract() [function.extract]: First argument should be an array
in **/home/servers/www.devshed.com-mambo/www/includes/templating.php** on line **34**

PHP RSS

# Working with Images and Text Flows in PDF Files with PHP 5

By: Alejandro Gervasio
Rating: ★★★★☆ / 2
2007-11-13





When it comes to developing database-driven web applications that deliver their contents to end users in a great variety of formats, PHP can be a killer scripting language. This becomes even more evident when these contents must be displayed through one or more PDF files, which have to be built dynamically before being sent to the client. So if you want to learn how to start building PDF documents in PHP 5, then you should begin reading this article now!

Welcome to the third article of the series that began with "Building PDF Documents with PHP 5." Composed of five approachable tutorials, this series takes you through using the most relevant methods that come packaged with the popular "PDFlib" library to create content-rich PDF files directly from inside your own PHP 5 scripts. It complements the corresponding theory with many educational code samples.

Now that you know the subject of this series, I'm going to spend some time refreshing the topics that I covered in the previous tutorial. This will help you to establish an appropriate continuity between it and the current one that you're hopefully reading with interest.

As you'll surely recall, in the second part of the series I showed you how to use a number of straightforward methods that come bundled with the aforementioned "PDFlib" package to build a few basic PDF documents that contained some basic texts. In this case, the text in question was distributed primitively across multiple lines by using a combination of the "set_text_pos()", "show()", and "continue_text()" methods respectively. This came in handy for performing some common tasks, such as specifying what X and Y coordinates should be used on the PDF document to display this text, and that it should be distributed in several chunks of strings.

Of course, due to the easy learning curve offered by many of the methods included with the PDFlib library, performing the tasks from a simple PHP 5 script shouldn't be a major concern for you, and should be tackled with minimal hassles.

So far, so good. At this point, you hopefully recall how to use the previous methods to display multiple lines of text in a concrete PDF file, so it's time to learn a few more method bundled with the library. As you'll see in the next few lines, these methods can be quite helpful for performing other tasks, like including images and blocks of texts into the PDF file in question.

So, are you ready to find out how to incorporate graphics and blocks of strings into a specific PDF document with PHP 5? Well, let's not waste more time in preliminaries and begin this educational journey now!

Before I proceed to show you how to include some basic images into a given PDF document, first I'd like you to recall properly how to use the handy "set_text_pos()" and "continue_text()" methods that I explained in the previous article of the series, in order to build PDF files that contain multiple lines of text.

Regarding the utilization of these methods, below I listed a pair of illustrative examples that show how to implement them. Here are the corresponding code samples, so have a look at them, please:

```
// example creating a basic PDF document with PHP and multiple
lines using the 'continue_text()' method

  try {

// create new instance of the 'PDFlib' class

  $pdf=new PDFlib();

// open new PDF file

  if(!$pdf->begin_document("",""){

throw new PDFlibException("Error creating PDF document. ".$pdf-
>get_errmsg());

}

  $pdf->set_info("Creator","example.php");

  $pdf->set_info("Author","Alejandro Gervasio");

  $pdf->set_info("Title","Example on using PHP to create PDF
docs");

  $pdf->begin_page_ext(421,595,"");
```

```
   $font=$pdf->load_font("Helvetica-Bold","winansi","");
  $pdf->setfont($font,24.0);
  $pdf->set_text_pos(50,500);
  $pdf->show("PHP is great for creating PDFs!");
  $pdf->continue_text('This is another line of text');
  $pdf->continue_text('This is another line of text');
  $pdf->continue_text('This is another line of text');
// end page
  $pdf->end_page_ext("");

// end document
  $pdf->end_document("");
// get buffer contents
    $buffer=$pdf->get_buffer();
// get length of buffer
  $len=strlen($buffer);
// display PDF document
  header("Content-type: application/pdf");
  header("Content-Length: $len");
  header("Content-Disposition: inline; filename=example.pdf");
   echo $buffer;
}
catch (PDFlibException $e){
  echo 'Error Number:'.$e->get_errnum()."n";
  echo 'Error Message:'.$e->get_errmsg();
 exit();
```

```
}

// example creating a basic PDF document with PHP and multiple
lines using the 'set_text_pos()' method

  try {

// create new instance of the 'PDFlib' class

   $pdf=new PDFlib();

// open new PDF file

  if(!$pdf->begin_document("",""){

throw new PDFlibException("Error creating PDF document. ".$pdf-
>get_errmsg());

}

  $pdf->set_info("Creator","example.php");

  $pdf->set_info("Author","Alejandro Gervasio");

  $pdf->set_info("Title","Example on using PHP to create PDF
docs");

  $pdf->begin_page_ext(421,595,"");



   $font=$pdf->load_font("Helvetica-Bold","winansi","");

  $pdf->setfont($font,24.0);

  $pdf->set_text_pos(50,500);

  $pdf->show("PHP is great for creating PDFs!");

  $pdf->set_text_pos(50,450);

  $pdf->show('This is another line of text');

  $pdf->set_text_pos(50,400);

  $pdf->show('This is another line of text');

  $pdf->set_text_pos(50,350);

  $pdf->show('This is another line of text');

// end page

  $pdf->end_page_ext("");
```

```
// end document

  $pdf->end_document("");

// get buffer contents

    $buffer=$pdf->get_buffer();

// get length of buffer

     $len=strlen($buffer);

// display PDF document

  header("Content-type: application/pdf");

  header("Content-Length: $len");

  header("Content-Disposition: inline; filename=example.pdf");

   echo $buffer;

}

    catch (PDFlibException $e){

  echo 'Error Number:'.$e->get_errnum()."n";

  echo 'Error Message:'.$e->get_errmsg();

     exit();

}
```

As you can see, the two examples above demonstrate how to utilize the "set_text_pos()", "show()", and "continue_text ()" methods that come integrated with the PDFlib library to build a couple of PDF documents that include some basic multi-line texts.

Of course, in this case, the "set_text_pos()" method makes it really easy to display a given string in a PDF file using a pair of X, Y coordinates, which can be expanded to scope two or more lines of text.

All right, at this stage you've hopefully recalled the complete sequence of steps required to build basic PDF files that contain simple strings. Considering this, I believe that it's time to continue learning more methods that come bundled with the PDFlib package.

In the upcoming section of this tutorial I'm going to show you how to incorporate a simple image into an existing PDF file to make it slightly more appealing. Of course, to see how this will be done, you have to click on the link below and keep reading.

According to the concepts that I deployed in the previous section, the PDFlib library provides PHP developers with some intuitive methods for including different types of images into a specific PDF file.

Initially, PHP 4 offered a decent variety of native methods aimed at working with images and PDF documents in

conjunction. Unfortunately, many of them have been deprecated. If you're still using this version of PHP and your php.ini file has been set up for working with PDF files, you're completely free to use these methods in accordance with your personal needs.

In this case, though, I'm going to use only two methods for including images into a PDF document, called "load_image ()" and "fit_image()" respectively, since they're pretty intuitive and easy to learn. However, you should take into account that these might not work as expected, depending on the respective versions of PHP and the "PDFlib" package that you have installed on your system. So be aware of this issue when building your PDF-related scripts.

Now that I have clarified that point, please look at the following example. It shows how to include a basic image into a specific PDF file. The pertinent code sample is as follows:

```
// example creating a basic PDF document and include a sample
image

  try {

// create new instance of the 'PDFlib' class

  $pdf=new PDFlib();

// open new PDF file

  if(!$pdf->begin_document("",""){

throw new PDFlibException("Error creating PDF document. ".$pdf-
>get_errmsg());

}

  $pdf->set_info("Creator","example.php");

  $pdf->set_info("Author","Alejandro Gervasio");

  $pdf->set_info("Title","Example on using PHP to create PDF
docs");

  $pdf->begin_page_ext(421,595,"");



   $font=$pdf->load_font("Helvetica-Bold","winansi","");

  $pdf->setfont($font,24.0);

  $pdf->set_text_pos(50,500);

  $pdf->show("PHP is great for creating PDFs!");

// load image

   $img=$pdf->load_image("jpeg","sample_image.jpg","");

// display image on page
```

```
  $pdf->fit_image($img,390,575,"");

// close image resource

  $pdf->close_image($img);

// end page

  $pdf->end_page_ext("");

// end document

  $pdf->end_document("");

// get buffer contents

  $buffer=$pdf->get_buffer();

// get length of buffer

    $len=strlen($buffer);

// display PDF document

  header("Content-type: application/pdf");

  header("Content-Length: $len");

  header("Content-Disposition: inline; filename=example.pdf");

   echo $buffer;

}

catch (PDFlibException $e){

  echo 'Error Number:'.$e->get_errnum()."n";

  echo 'Error Message:'.$e->get_errmsg();

 exit();

}
```

As shown in the prior example, including a simple image into a specific PDF file is reduced to loading the pertinent graphic via the brand new "load_image()" method, and then displaying it using another method, named "fit_image()". As you might have guessed, the first method comes in handy for loading images of different types (JPEG, GIF, PNG, etc.) and the second one simply shows it at a specified position within the document in question. Quite simple, isn't it?

Besides, as I explained earlier, the PDFlib library comes equipped with some additional methods that can be used to display images on a given PDF file, but in my opinion the ones shown here are the most intuitive. If you're interested in learning how to use these additional methods, please visit the PHP official web site and read the PDF-related section.

So far, so good. At this time you've hopefully grasped the logic required to display some basic images on a concrete PDF document, which indeed is a no-brainer process that can be tackled with minimal hassles. So assuming that you already learned this topic, let's move forward and see how to use some other useful methods that come integrated with the PDFlib library to display blocks of texts, called "text flows."

Sounds quite interesting, doesn't it? So go ahead and read the next section. I'll be there, waiting for you.

As I stated in the section that you just read, the PDFlib package provided PHP developers with a bunch of handy methods aimed at displaying in a specific PDF document, several blocks of text, called "text flows."

Basically, these blocks can be easily placed at any position within the pertinent document. It's also possible to specify their respective width and height values, which can be useful if you want to include strings that have predefined dimensions.

However, let me put aside this boring theory for a moment and show you a practical example of how to include a basic text flow into a primitive PDF file. The corresponding code sample is listed below, so have a close look at it:

```
try{

// example creating a basic PDF document and display text flow

// create new instance of the 'PDFlib' class

  $pdf=new PDFlib();

// open new PDF file

      if(!$pdf->begin_document("","")){

throw new PDFlibException("Error creating PDF document. ".$pdf-
>get_errmsg());

}

  $pdf->set_info("Creator","example.php");

  $pdf->set_info("Author","Alejandro Gervasio");

  $pdf->set_info("Title","Example on using PHP to create PDF
docs");

  $pdf->begin_page_ext(421,595,"");



   $font=$pdf->load_font("Helvetica-Bold","winansi","");

  $pdf->setfont($font,24.0);

  $pdf->set_text_pos(50,500);

  $pdf->show("PHP is great for creating PDFs!");

// create text flow
```

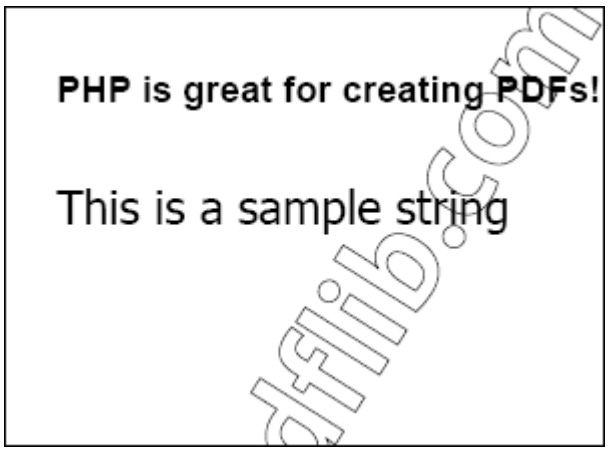
```
$textflow=$pdf->create_textflow('This is a sample
string','fontname=Tahoma fontsize=30 encoding=winansi');

// display text flow

  $pdf->fit_textflow($textflow,50,450,400,220,'');

// delete text flow

  $pdf->delete_textflow($textflow);

// end page

  $pdf->end_page_ext("");

// end document

  $pdf->end_document("");

// get buffer contents

  $buffer=$pdf->get_buffer();

// get length of buffer

    $len=strlen($buffer);

// display PDF document

  header("Content-type: application/pdf");

  header("Content-Length: $len");

  header("Content-Disposition: inline; filename=example.pdf");

    echo $buffer;

}

catch (PDFlibException $e){

  echo 'Error Number:'.$e->get_errnum()."n";

  echo 'Error Message:'.$e->get_errmsg();

 exit();

}
```

As you can see, the above example uses three new methods, called "create_text_flow()", "fit_text_flow()", and "delete_text_flow()" to build a simple block of text, and then display this block at a specified position within the PDF file, and finally delete it from the respective buffer.

In addition, it's worth noting here how the "fit_text_flow()" method used in this case to specify the respective width and

height values should be assigned to the text flow just created.

And finally, to extend the explanation of including a basic text flow into a given PDF file, below I included a screen shot that shows the output generated by the previous script:




All right, having demonstrated how to include one basic text flow into a simple PDF file using the PDFlib package in conjunction with PHP 5, I hope that you'll have a strong enough foundation to start developing your own testing examples, with the purpose of acquiring a solid background in building PDF documents with PHP.

**Final thoughts**

That's all for the moment. In this third part of the series I showed you how to use a group of additional methods bundled with the "PDFlib" library, how to include images into a given PDF file, and introduced simple blocks of texts, called "text flows."

In the upcoming article, I'm going to teach you how to display a few basic shapes within a PDF file, which can be useful if you want to build PDF documents that contain richer contents.

To learn more about these interesting topics, don't miss the next part of the series!