61. FPDF LIBRARY.

61.1. What is FPDF?

FPDF is a PHP class which allows to generate PDF files with pure PHP, that is to say without using the PDFlib library. F from FPDF stands for Free: you may use it for any kind of usage and modify it to suit your needs.

FPDF has other advantages: high level functions. Here is a list of its main features:

- Choice of measure unit, page format and margins.
- Page header and footer management.
- Automatic page break.
- Automatic line break and text justification.
- Image support (JPEG, PNG and GIF).
- Colors.
- Links.
- TrueType, Type1 and encoding support.
- Page compression.

FPDF requires no extension (except ***zlib*** to activate compression and ***GD*** for GIF support). It works with PHP 4 and PHP 5 (the latest version requires at least PHP 4.3.10).

What languages can I use? The class can produce documents in many languages other than the Western European ones: Central European, Cyrillic, Greek, Baltic and Thai, provided you own TrueType or Type1 fonts with the desired character set. Chinese, Japanese and Korean are supported too.

UTF-8 support is also available.

What about performance? Of course, ***the generation speed of the document is less than with PDFlib***. However, the performance penalty keeps very reasonable and suits in most cases, unless your documents are particularly complex or heavy.

61.2. Minimal Example.

Let's start with the classic example:

```
<?php
require('fpdf.php');

$pdf = new FPDF();
$pdf->AddPage();
$pdf->SetFont('Arial','B',16);
$pdf->Cell(40,10,'Hello World!');
$pdf->Output();
?>
```

After including the library file, we create an FPDF object. The FPDF() constructor is used here with the default values: pages are in A4 portrait and the unit of measure is millimeter. It could have been specified explicitly with:

```
$pdf = new FPDF('P','mm','A4');
```

It's possible to use landscape (**L**), other page sizes (such as **Letter** and **Legal**) and units (**pt**, **cm**, **in**).

There's no page at the moment, so we have to add one with *AddPage*(). The origin is at the upper-left corner and the current position is by default set at 1 cm from the borders; the margins can be changed with *SetMargins*().

Before we can print text, it's mandatory to select a font with *SetFont*(), otherwise the document would be invalid. We choose ***Arial bold 16***:

```
$pdf->SetFont('Arial','B',16);
```

We could have specified italics with **I**, underlined with **U** or a regular font with an empty string (or any combination). Note that the font size is given in points, not millimeters (or another user unit); it's the only exception. The other standard fonts are Times, Courier, Symbol and ZapfDingbats.

We can now print a cell with *Cell*(). A cell is a rectangular area, possibly framed, which contains a line of text. It is output at the current position. We specify its dimensions, its text (centered or aligned), if borders should be drawn, and where the current position moves after it (to the right, below or to the beginning of the next line). To add a frame, we would do this:

```
$pdf->Cell(40,10,'Hello World !',1);
```

To add a new cell next to it with centered text and go to the next line, we would do:

```
$pdf->Cell(60,10,'Powered by FPDF.',0,1,'C');
```

Remark: the line break can also be done with *Ln*(). This method additionally allows to specify the height of the break.

Finally, the document is closed and sent to the browser with *Output*(). We could have saved it to a file by passing the desired file name.

Caution: in case when the PDF is sent to the browser, nothing else must be output by the script, neither before nor after (no HTML, not even a space or a carriage return). If you send something before, you will get the error message: "***Some data has already been output, can't send PDF file***". If you send something after, the document might not display.

61.3. Header, footer, page break and image.

Here's a two page example with header, footer and logo:

```
<?php
require('fpdf.php');

class PDF extends FPDF
{
// Page header
function Header()
{
    // Logo
    $this->Image('logo.png',10,6,30);
    // Arial bold 15
    $this->SetFont('Arial','B',15);
    // Move to the right
    $this->Cell(80);
    // Title
    $this->Cell(30,10,'Title',1,0,'C');
    // Line break
    $this->Ln(20);
}

// Page footer
function Footer()
{
    // Position at 1.5 cm from bottom
    $this->SetY(-15);
    // Arial italic 8
    $this->SetFont('Arial','I',8);
    // Page number
    $this->Cell(0,10,'Page '.$this->PageNo().'/{nb}',0,0,'C');
}
}
```

```
// Instanciation of inherited class
$pdf = new PDF();
$pdf->AliasNbPages();
$pdf->AddPage();
$pdf->SetFont('Times','',12);
for($i=1;$i<=40;$i++)
    $pdf->Cell(0,10,'Printing line number '.$i,0,1);
$pdf->Output();
?>
```

This example makes use of the *Header*() and *Footer*() methods to process page headers and footers. They are called automatically. They already exist in the FPDF class but do nothing, therefore we have to extend the class and override them.

The logo is printed with the ***Image***() method by specifying its upper-left corner and its width. The height is calculated automatically to respect the image proportions.

To print the page number, a null value is passed as the cell width. It means that the cell should extend up to the right margin of the page; this is handy to center text. The current page number is returned by the ***PageNo***() method; as for the total number of pages, it's obtained via the special value ***{nb}*** which is substituted when the document is finished (provided you first called ***AliasNbPages***()).

Note the use of the ***SetY***() method which allows to set position at an absolute location in the page, starting from the top or the bottom.

Another interesting feature is used here: the automatic page breaking. As soon as a cell would cross a limit in the page (at 2 centimeters from the bottom by default), a break is issued and the font restored. Although the header and footer select their own font (*Arial*), the body continues with Times. This mechanism of automatic restoration also applies to colors and line width. The limit which triggers page breaks can be set with ***SetAutoPageBreak***().

61.4. Header, footer, page break and image.

Let's continue with an example which prints justified paragraphs. It also illustrates the use of colors.

```
<?php
require('fpdf.php');

class PDF extends FPDF
{
function Header()
{
    global $title;

    // Arial bold 15
    $this->SetFont('Arial','B',15);
    // Calculate width of title and position
    $w = $this->GetStringWidth($title)+6;
    $this->SetX((210-$w)/2);
    // Colors of frame, background and text
    $this->SetDrawColor(0,80,180);
    $this->SetFillColor(230,230,0);
    $this->SetTextColor(220,50,50);
    // Thickness of frame (1 mm)
    $this->SetLineWidth(1);
    // Title
    $this->Cell($w,9,$title,1,1,'C',true);
    // Line break
    $this->Ln(10);
}

function Footer()
{
    // Position at 1.5 cm from bottom
    $this->SetY(-15);
    // Arial italic 8
```

```
        $this->SetFont('Arial','I',8);
        // Text color in gray
        $this->SetTextColor(128);
        // Page number
        $this->Cell(0,10,'Page '.$this->PageNo(),0,0,'C');
    }

    function ChapterTitle($num, $label)
    {
        // Arial 12
        $this->SetFont('Arial','',12);
        // Background color
        $this->SetFillColor(200,220,255);
        // Title
        $this->Cell(0,6,"Chapter $num : $label",0,1,'L',true);
        // Line break
        $this->Ln(4);
    }

    function ChapterBody($file)
    {
        // Read text file
        $txt = file_get_contents($file);
        // Times 12
        $this->SetFont('Times','',12);
        // Output justified text
        $this->MultiCell(0,5,$txt);
        // Line break
        $this->Ln();
        // Mention in italics
        $this->SetFont('','I');
        $this->Cell(0,5,'(end of excerpt)');
    }

    function PrintChapter($num, $title, $file)
    {
```

```
        $this->AddPage();
        $this->ChapterTitle($num,$title);
        $this->ChapterBody($file);
    }
    }

    $pdf = new PDF();
    $title = '20000 Leagues Under the Seas';
    $pdf->SetTitle($title);
    $pdf->SetAuthor('Jules Verne');
    $pdf->PrintChapter(1,'A RUNAWAY REEF','20k_c1.txt');
    $pdf->PrintChapter(2,'THE PROS AND CONS','20k_c2.txt');
    $pdf->Output();
    ?>
```

The ***GetStringWidth***() method allows to determine the length of a string in the current font, which is used here to calculate the position and the width of the frame surrounding the title. Then colors are set (via *SetDrawColor(), SetFillColor()* and *SetTextColor()*) and the thickness of the line is set to 1 mm (instead of 0.2 by default) with *SetLineWidth*(). Finally, we output the cell (the last parameter true indicates that the background must be filled).

The method used to print the paragraphs is *MultiCell*(). Each time a line reaches the right extremity of the cell or a carriage return character is met, a line break is issued and a new cell automatically created under the current one. Text is justified by default.

Two document properties are defined: the title (*SetTitle*()) and the author (*SetAuthor*()). There are several ways to view them in Adobe Reader. The first one is to open the file directly with the reader, go to the **File** menu and choose the **Properties** option. The second one, also available from the plug-in, is to right-click and select **Document Properties**. The third method is to type the ***Ctrl+D*** key combination.

61.5. Multi-columns.

This example is a variant of the previous one showing how to lay the text across multiple columns.

```
<?php
require('fpdf.php');

class PDF extends FPDF
{
// Current column
var $col = 0;
// Ordinate of column start
var $y0;

function Header()
{
    // Page header
    global $title;

    $this->SetFont('Arial','B',15);
    $w = $this->GetStringWidth($title)+6;
    $this->SetX((210-$w)/2);
    $this->SetDrawColor(0,80,180);
    $this->SetFillColor(230,230,0);
    $this->SetTextColor(220,50,50);
    $this->SetLineWidth(1);
    $this->Cell($w,9,$title,1,1,'C',true);
    $this->Ln(10);
    // Save ordinate
    $this->y0 = $this->GetY();
}

function Footer()
{
    // Page footer
```

```
        $this->SetY(-15);
        $this->SetFont('Arial','I',8);
        $this->SetTextColor(128);
        $this->Cell(0,10,'Page '.$this->PageNo(),0,0,'C');
    }

    function SetCol($col)
    {
        // Set position at a given column
        $this->col = $col;
        $x = 10+$col*65;
        $this->SetLeftMargin($x);
        $this->SetX($x);
    }

    function AcceptPageBreak()
    {
        // Method accepting or not automatic page break
        if($this->col<2)
        {
            // Go to next column
            $this->SetCol($this->col+1);
            // Set ordinate to top
            $this->SetY($this->y0);
            // Keep on page
            return false;
        }
        else
        {
            // Go back to first column
            $this->SetCol(0);
            // Page break
            return true;
        }
    }
```

```
    function ChapterTitle($num, $label)
    {
        // Title
        $this->SetFont('Arial','',12);
        $this->SetFillColor(200,220,255);
        $this->Cell(0,6,"Chapter $num : $label",0,1,'L',true);
        $this->Ln(4);
        // Save ordinate
        $this->y0 = $this->GetY();
    }

    function ChapterBody($file)
    {
        // Read text file
        $txt = file_get_contents($file);
        // Font
        $this->SetFont('Times','',12);
        // Output text in a 6 cm width column
        $this->MultiCell(60,5,$txt);
        $this->Ln();
        // Mention
        $this->SetFont('','I');
        $this->Cell(0,5,'(end of excerpt)');
        // Go back to first column
        $this->SetCol(0);
    }

    function PrintChapter($num, $title, $file)
    {
        // Add chapter
        $this->AddPage();
        $this->ChapterTitle($num,$title);
        $this->ChapterBody($file);
    }
    }
```

```
$pdf = new PDF();
$title = '20000 Leagues Under the Seas';
$pdf->SetTitle($title);
$pdf->SetAuthor('Jules Verne');
$pdf->PrintChapter(1,'A RUNAWAY REEF','20k_c1.txt');
$pdf->PrintChapter(2,'THE PROS AND CONS','20k_c2.txt');
$pdf->Output();
?>
```

The key method used is ***AcceptPageBreak***(). It allows to accept or not an automatic page break. By refusing it and altering the margin and current position, the desired column layout is achieved.

For the rest, not many changes; two properties have been added to the class to save the current column number and the position where columns begin, and the ***MultiCell***() call specifies a 6 centimeter width.

61.6. Tables.

This tutorial shows different ways to make tables.

```
<?php
require('fpdf.php');

class PDF extends FPDF
{
// Load data
function LoadData($file)
{
    // Read file lines
    $lines = file($file);
    $data = array();
    foreach($lines as $line)
        $data[] = explode(';',trim($line));
    return $data;
}

// Simple table
function BasicTable($header, $data)
{
    // Header
    foreach($header as $col)
        $this->Cell(40,7,$col,1);
    $this->Ln();
    // Data
    foreach($data as $row)
    {
        foreach($row as $col)
            $this->Cell(40,6,$col,1);
        $this->Ln();
    }
}
```

```
// Better table
function ImprovedTable($header, $data)
{
    // Column widths
    $w = array(40, 35, 40, 45);
    // Header
    for($i=0;$i<count($header);$i++)
        $this->Cell($w[$i],7,$header[$i],1,0,'C');
    $this->Ln();
    // Data
    foreach($data as $row)
    {
        $this->Cell($w[0],6,$row[0],'LR');
        $this->Cell($w[1],6,$row[1],'LR');
        $this->Cell($w[2],6,number_format($row[2]),'LR',0,'R');
        $this->Cell($w[3],6,number_format($row[3]),'LR',0,'R');
        $this->Ln();
    }
    // Closing line
    $this->Cell(array_sum($w),0,'','T');
}

// Colored table
function FancyTable($header, $data)
{
    // Colors, line width and bold font
    $this->SetFillColor(255,0,0);
    $this->SetTextColor(255);
    $this->SetDrawColor(128,0,0);
    $this->SetLineWidth(.3);
    $this->SetFont('','B');
    // Header
    $w = array(40, 35, 40, 45);
    for($i=0;$i<count($header);$i++)
        $this->Cell($w[$i],7,$header[$i],1,0,'C',true);
    $this->Ln();
```

```
        // Color and font restoration
        $this->SetFillColor(224,235,255);
        $this->SetTextColor(0);
        $this->SetFont('');
        // Data
        $fill = false;
        foreach($data as $row)
        {
            $this->Cell($w[0],6,$row[0],'LR',0,'L',$fill);
            $this->Cell($w[1],6,$row[1],'LR',0,'L',$fill);
                $this->Cell($w[2],6,number_format($row[2]),'LR',0,'R',
$fill);
                $this->Cell($w[3],6,number_format($row[3]),'LR',0,'R',
$fill);
            $this->Ln();
            $fill = !$fill;
        }
        // Closing line
        $this->Cell(array_sum($w),0,'','T');
    }
    }

    $pdf = new PDF();
    // Column headings
    $header = array('Country', 'Capital', 'Area (sq km)', 'Pop.
(thousands)');
    // Data loading
    $data = $pdf->LoadData('countries.txt');
    $pdf->SetFont('Arial','',14);
    $pdf->AddPage();
    $pdf->BasicTable($header,$data);
    $pdf->AddPage();
    $pdf->ImprovedTable($header,$data);
    $pdf->AddPage();
    $pdf->FancyTable($header,$data);
    $pdf->Output();
    ?>
```

A table being just a collection of cells, it's natural to build one from them. The first example is achieved in the most basic way possible: simple framed cells, all of the same size and left aligned. The result is rudimentary but very quick to obtain.

The second table brings some improvements: each column has its own width, headings are centered, and numbers right aligned. Moreover, horizontal lines have been removed. This is done by means of the border parameter of the ***Cell***() method, which specifies which sides of the cell must be drawn. Here we want the left (L) and right (R) ones. It remains the problem of the horizontal line to finish the table. There are two possibilities: either check for the last line in the loop, in which case we use **LRB** for the border parameter; or, as done here, add the line once the loop is over.

The third table is similar to the second one but uses colors. Fill, text and line colors are simply specified. Alternate coloring for rows is obtained by using alternatively transparent and filled cells.