Tema 09 – Generación de Archivos PDF

2º DAW – Desarrollo Web Entorno Servidor

Profesor Juan Carlos Moreno

CURSO 2024/2025

Tabla de Contenido

9 Generación Archivos PDF

9.1 Introducción

Hay muchas situaciones en las que crear un documento PDF (Portable Documento File) es una solución efectiva de comunicación y apropiada para muchos de nuestros proyectos.

Existen muchas clases gratuitas PDF o también librerías PDFlib de las que unas son comerciales y otras gratuitas. A continuación veremos las más populares clases y librerías PHP:

- FPDF. fue el primer script PHP para la creación de archivos PDF de forma dinámica. Es el que nosotros vamos a utilizar
- TCPDF. Este proyecto ea un script para generación de archivos PDF basado en la librería FPDF. Inicialmente fue una versión para PHP5 con soporte UTF-8 pero se ha convertido en un proyecto independiente el cual tiene gran soporte para imágenes, incluso para colores CMYK
- Zend_Pdf. El framework Zend tiene una clase para la generación de archivos PDF, la cual tiene menos funciones que FPDF y otras clases.
- PDFlib. La librería PDFlib es la solución comercial para la generación de archivos PDF con PHP. Cuando se instala (compilada), puede empezar a usarse de forma inmediata en el código PHP sin necesidad de incluir otras clases o librerías. Es una librería bastante buena y rápida aunque su costo es un poco alto.

9.2 Clase FPDF

FPDF es una clase escrita en PHP que permite generar documentos PDF directamente desde PHP, es decir, sin usar la biblioteca PDFlib. La F de FPDF significa Free (gratis y libre): puede usted usarla para cualquier propósito y modificarla a su gusto para satisfacer sus necesidades.

Sus principales características son:

- Elección de la unidad de medida, formato de página y márgenes
- Gestión de cabeceras y pies de página
- Salto de página automático
- Salto de línea y justificación del texto automáticos
- Admisión de imágenes (JPEG, PNG y GIF)
- Colores
- Enlaces
- Admisión de fuentes TrueType, Type1 y codificación
- Compresión de página

9.2.1 Instalación

Esta clase requiere al menos PHP 4 o superior. Además tiene algunas extensiones que nos pueden resultar útiles.

Después de esta pequeña introducción vamos a pasar a descargar e instalar las librerías necesarias para su utilización. Para ello nos descargamos la última versión de FPDF en el siguiente enlace http://www.fpdf.org/, en la sección de descargas.

Una vez descargado lo subimos a nuestro servidor y lo colocamos en una carpeta llamada fpdf en la raíz del dominio, o cualquier otro directorio que deseemos, con tal que nos acordemos dónde la hemos puesto y la incluyamos correctamente en los scripts donde pensemos utilizarla.

Por ejemplo si la hemos instalado en la carpeta fpdf de nuestro script php en la cabecera pondremos

```
require ('fpdf/fpdf.php');
```

9.2.2 Ejemplo Básico

FPDF utiliza como cualquier clase de programación orientada a objetos con PHP, por lo que lo más importante será conocer y dominar sus diferentes métodos y propiedades. Vamos a empezar por un ejemplo muy sencillo que no requiere mucha programación PHP.

El formato general para escribir una pagina en PDF sería el siguiente:

```
<?php
	require('fpdf/fpdf.php');
	$pdf=new FPDF();
	$pdf->AddPage();
	$pdf->SetFont('Arial','B',16);
	$pdf->Cell(40,10,'¡Mi primera página pdf con FPDF!');
	$pdf->Output();
?>
```

Analizando todas las líneas:

- Lo primero que hacemos es incluir la librería fpdf.php
- En la linea $pdf=new FPDF(); lo que estamos haciendo es crear el objeto FPDF. Si no ponemos nada entre los paréntesis, el objeto se creará con los valores por defecto, en este caso serian los siguientes: el tamaño de pagina es A4, el formato alargado y la unidad de medida el milímetro. Si queremos modificar estos parámetros seria en el siguiente orden $pdf=new FPDF('formato','unidad de medida','tamaño');
- En la linea $pfd->AddPage(); añadimos una página.
- Con SetFont(); le damos formato al texto diciendo el tipo de letra, si es en negrita o no, y el tamaño de la letra.
- Ya en la linea $pdf->Cell(); empezamos a escribir el contenido de la página. Empezamos diciendo el ancho de la celda donde vamos a escribir, el alto de la celda, y el contenido de la celda. Tiene algunos parámetros más que iremos explicando detenidamente en los siguientes artículos.
- La ultima linea $pdf->Output(); lo que hace es cerrar el archivo y enviarlo al navegador. Es importante no poner esta linea antes de terminar de escribir el archivo ya que nos dará error. Además si no lo pones justo al final y escribes algunas lineas más de código no relacionado con el PDF puede aparecerte el documento en blanco.

9.2.3 Métodos Básicos de la clase FPDF.

9.2.3.1 Fpdf()

Vamos a empezar con el método constructor FPDF() que nos permite crear el documento pdf y darle un formato. Los parámetros que le pasemos se usarán en todos los métodos. Su sintaxis es la siguiente:

FPDF([string orientación [, string unidad [, mixed formato]]);

Parámetros:

- Orientación es la forma de colocación de la página, es decir, debemos indicar si es normal o apaisada. El valor por defecto Pes normal. El valor para apaisada es L
- Unidad es la medida de usuario y sus posibles valores son: pt punto, mm milímetro, cm centímetro e in pulgada. El valor por defecto es el mm
- Formato de la página. Puede tener los siguientes valores: A3, A4, A5, Letter y Legal. El valor por defecto es A4

```
$pdf=new FPDF('L','pt','Legal');
$pdf = new FPDF();
```

9.2.3.2 AddPage()

Esta función nos añade una página nueva al documento pdf. Como parámetros tan solo tiene la orientación y el formato, el resto de características las coge por defecto del constructor.

Su sintaxis es la siguiente:

```
$pdf->AddPage([string orientacion[,mixed formato]]);
```

Los parámetros orientación y formato son iguales que en FPDF(). Si no le pasas parámetros cogerá automáticamente los del constructor.

Veamos el ejemplo más típico:

```
$pdf->AddPage();
```

9.2.3.3 SetFont()

Es la función que nos permite establecer el formato de la fuente utilizada en el archivo pdf. Es obligatorio llamar a esta función al inicio de la creación del archivo ya que sino el documento no sería valido.

Si queremos añadir un tipo de fuente que no está en el standar debemos utilizar la función AddFont(); que veremos más adelante.

La sintaxis de SetFont es la siguiente:

```
SetFont(string familia[, string estilo [, float size]]);
```

- familia: familia de fuente que pueden ser las standars(Courier,Helvetica o Arial, Times, Symbol,ZapfDingbats) o añadir una mediante AddFont();
- estilo: estilo de la fuente que puede ser regular, negrita B, italica I y subindice U.
- size: tamaño de la fuente en puntos. Su valor por defecto es 12.

Un ejemplo seria el siguiente:

```
SetFont('Helvetica','I',13);
```

9.2.3.4 Cell()

Esta función nos imprime una celda donde vamos a imprimir nuestro texto. Tiene bordes opcionales y color de fondo. En esta celda podemos colocar el texto alineado o centrado.

Su sintaxis es la siguiente:

```
Cell(float w [, float h [, string texto [, mixed borde [, int ln [, string
align [, boolean fill [, mixed link]]]]]]])
```

- w: ancho de la celda. Si ponemos 0 la celda se extiende hasta el margen derecho.
- H: alto de la celda.
- Texto: el texto que le vamos a añadir.
- Borde: nos dice si van a ser visibles o no. si es 0 no serán visibles, si es 1 se verán los bordes.
- Ln: nos dice donde se empezara a escribir después de llamar a esta función. Siendo 0 a la derecha, 1 al comienzo de la siguiente línea, 2 debajo.
- Align: para alinear el texto. L alineado a la izquierda, C centrado y R alineado a la derecha.
- Fill: nos dice si el fondo de la celda va a ir con color o no. los valores son True o False

Un ejemplo sería el siguiente:

```
$pdf->Cell(40,10,'¡Hola, Mundo!',1);
$pdf->Cell(60,10,'Hecho con FPDF.',0,1,'C');
```

Añadimos una nueva celda con el texto centrado e ir a la siguiente línea.

9.2.3.5 Output()

Nos envía el documento al navegador, a un fichero local o a una cadena. Podemos abrirlo en un cuadro de diálogo o prepararlo para una descarga.

Su sintaxis es la siguiente:

string Output([string nombre, string destino])

- nombre: damos nombre al fichero, si no se indica lo llama por defecto doc.pdf
- destino: destino de envío en el documento. I envía el fichero al navegador con la opción de guardar como..., D envía el documento al navegador preparado para la descarga, F guarda el fichero en un archivo local, S devuelve el documento como una cadena.

Un ejemplo seria el siguiente:

```
$fpdf->Output('prueba','I');
```

9.2.3.6 Cabecera y Pié de página

Para editar la cabecera y el pié de página necesitamos heredar la clase FPDF y sobrescribir los métodos Header() y Footer().

9.2.3.6.1 Cabecera

Vamos a comenzar con la cabecera de nuestro archivo pdf. Lo que vamos a hacer es colocarle un logo y una serie de estilos.

Veamos el siguiente ejemplo.

```
<?
require('fpdf.php');
class PDF extends FPDF
{
   //Cabecera de página
   function Header()
   {

      $this->Image('logo.png',10,8,33);
      $this->SetFont('Arial','B',12);
      $this->Cell(30,10,'Title',1,0,'C');
   }
}

//Creación del objeto de la clase heredada
$pdf=new PDF();
$pdf->AddPage();
$pdf->SetFont('Times','',12);
//Aquí escribimos lo que deseamos mostrar...
$pdf->Output();
?>
```

La línea $this->Image ('logo.png',12,9,30); nos coloca la imagen que le pasamos por parámetro. En esta función pasamos como parámetro el archivo donde se encuentra la imagen, la abscisa de la esquina superior izquierda, ordenada de la esquina superior izquierda y la anchura de la imagen. Podemos pasar más parámetros pero ya lo comentaremos más adelante.

En la siguiente línea ($this->SetFont('Arial','B',12);) damos formato al texto de la cabecera.

La línea $this->Cell (30,10,'Title',1,0,'C'); nos imprime el Título del documento.

Una vez que hemos terminado de sobrescribir el método Header tan sólo nos queda crear el objeto de la clase heredada y realizar nuestro documento pdf.

9.2.3.6.2 Pié de Página

Para dar formato al pie tenemos que realizar la misma operación que con la cabecera, es decir, sobrescribir el método Footer.

Con ello el documento completo quedaría de la siguiente forma:

```
<?
require('fpdf.php');
class PDF extends FPDF
{
   //Cabecera de página
   function Header()
```

```
    {

        $this->Image('logo.png',10,8,33);
        $this->SetFont('Arial','B',12);
        $this->Cell(30,10,'Title',1,0,'C');
    }

//Pie de página
function Footer()
{

        $this->SetY(-10);
        $this->SetFont('Arial','I',8);
        $this->Cell(0,10,'Page '.$this->PageNo().'/{nb}',0,0,'C');
    }
}

//Creación del objeto de la clase heredada
$pdf=new PDF();
$pdf->AddPage();
$pdf->SetFont('Times','',12);
//Aquí escribimos lo que deseamos mostrar...
$pdf->Output();
?>
```

La línea $this->SetY (-15); nos posiciona el pie a 1,0 cm del final.

La siguiente línea nos da el formato del texto del pie.

La última línea del método nos imprime el número de página actual. Con un margen que se extiende hasta el margen de la derecha, un alto de celda de 10 (el formato de medida depende del dado al inicio), un texto similar a esto Page 2, sin borde, con salto de línea a la derecha y el texto centrado.

Con esto tendríamos maquetados nuestra cabecera y pie de pagina en pdf. Aunque cabe destacar que podemos realizar más operaciones en la cabecera y pie, como por ejemplo darle fondo a las celdas, rellenarlas de colores, etc.

9.2.4 Insertar Imágenes con Image().

Esta función la utilizamos para añadir imágenes a nuestros archivos PDF.

Admite los formatos JPEG, PNG y GIF (para este formato necesitamos la extensión GD).

El formato de la imagen se puede especificar explícitamente o simplemente ser deducido a partir de la extensión del fichero.

Tenemos tres opciones en cuanto a la especificación de tamaño de la imagen:

Podemos especificar el ancho y el largo con unidades de medida definidas por nosotros mismos

1. Podemos especificar solo el ancho y el sistema calculará el alto automáticamente
2. No especificar nada, lo que hará que se imprima la imagen a 72 puntos por pulgada
3. Esta función además nos permite asociar un enlace a la imagen.

Por otro lado si repetimos las imágenes, FPDF solo guardará una copia para así bajar el peso del archivo.

Su sintaxis es la siguiente:

```
Image(string file [, float x [, float y [, float w [, float h [, string type
[, mixed link]]]]]])
```

Donde:

- file: nombre del archivo de la imagen.
- x: Abscisa de la esquina superior izquierda. Si no se especifica se utilizará la abscisa actual.
- y: Ordenada de la esquina superior izquierda. Si no se especifica se utilizará la ordenada actual.
- w: Ancho de la imagen en la página.
- h: Alto de la imagen en la página.
- type:Formato de la imagen.
- link: identificador devuelto por el método AddLink() o la url del enlace.

Un ejemplo sencillo sería el siguiente:

```
$this->Image('logo.jpg',10,8,22);
```

En este ejemplo nos calcularía el alto de la imagen de forma automática.

Un ejemplo completo lo haríamos asi:

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

$pdf=new FPDF();
//Primera página
$pdf->AddPage();
$pdf->SetFont('Arial','',15);
$pdf->Cell(40,20);
$pdf->Write(5,'A continuación mostramos una imagen ');
$pdf->Image('leon.jpg' , 80 ,22, 35 , 38,'JPG',
'http://www.ieslosremedios.org');

$pdf->Output();
?>
```

Este ejemplo es bastante sencillo y lo único que hace es mostrar un texto que nos presenta una imagen que aparecerá debajo centrada.

9.2.5 Mostrar un archivo txt en PDF

Expondremos un ejemplo, que luego iremos explicándolo paso a paso;

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

class PDF extends FPDF
{
```

```
//Cabecera de página
function Header()
{
	//Logo
	$this->Image("leon.jpg" , 10 ,8, 35 , 38 , "JPG" );
	//Arial bold 15
	$this->SetFont('Arial','B',15);
	//Movernos a la derecha
	$this->Cell(80);
	//Título
	$this->Cell(60,10,'Titulo del archivo',1,0,'C');
	//Salto de línea
	$this->Ln(20);

}

//Pie de página
function Footer()
{
	//Posición: a 1,5 cm del final
	$this->SetY(-15);
	//Arial italic 8
	$this->SetFont('Arial','I',8);
	//Número de página
	$this->Cell(0,10,'Page '.$this->PageNo(),0,0,'C');
}

function TituloArchivo($num,$label)
{
	$this->SetY(55);
	//Arial 12
	$this->SetFont('Arial','',12);
	//Color de fondo
	$this->SetFillColor(200,220,255);
	//Título
	$this->Cell(0,6,"Archivo $num : $label",0,1,'L',true);
	//Salto de línea
	$this->Ln(4);
}

function CuerpoArchivo($file)
{
	//Leemos el fichero
	$f=fopen($file,'r');
	$txt=fread($f,filesize($file));
	fclose($f);
	//Times 12
	$this->SetFont('Times','',12);
	//Imprimimos el texto justificado
	$this->MultiCell(0,5,$txt);
	//Salto de línea
	$this->Ln();

}

function ImprimirArchivo($num,$title,$file)
{
	$this->AddPage();
	$this->TituloArchivo($num,$title);
```

```
        $this->CuerpoArchivo($file);
}
}

$pdf=new PDF();
$title='Mostramos un archivo txt';
$pdf->SetTitle($title);
$pdf->SetY(65);
$pdf->ImprimirArchivo(1,'Archivo de prueba ','prueba1.txt');
$pdf->ImprimirArchivo(2,'Otro archivo','prueba2.txt');
$pdf->Output();
?>
```

TituloArchivo(), que nos permite especificar el diseño de los títulos de los archivos que vamos a mostrar. Le estamos dando un tipo de letra Arial con un tamaño 12, un color de fondo azul utilizando la función SetFillColor y colocando el titulo con su correspondiente formato, utilizando la función Cell.

A continuación creamos la función CuerpoArchivo() que es la que va a leer el archivo y imprimirlo en nuestro PDF.

Le pasamos como parámetro la ruta del archivo. Con fopen abrimos el archivo en modo lectura, después lo leemos con fread() y guardamos en una variable, cerramos el archivo y damos formato al texto que vamos a mostrar en nuestro PDF.

Para finalizar creamos la función ImprimirArchivo() que engloba las dos anteriores y nos hace más sencillo nuestro código.

Ya lo único que nos queda es lo de siempre, abrir nuestro archivo pdf, ponerle titulo y utilizar la función de ImprimirArchivo() las veces que queramos. Por ultimo cerrar el archivo y listo.

A continuación vamos a mostrar las dos funciones nuevas que hemos utilizado para la realización de este código:

9.2.5.1.1 Función SetFillColor()

Esta función nos define el color de relleno para celdas y rectángulos rellenos. Podemos expresar dicho color en RGB o escala de grises

Su sintaxis es la siguiente:

```
SetFillColor(int r [, int g, int b]);
```

siendo:

- r: si g y b los ponemos, nos muestra el rojo sino la escala de gris
- g: el verde
- b: azul

9.2.5.1.2 Función MultiCell()

Este método nos permite imprimir texto con saltos de línea. Estos pueden ser automáticos o explícito (con el carácter n).

Su sintaxis es la siguiente:

```
MultiCell(float w, float h, string txt [, mixed border [, string align [,
boolean fill]]])
```

Siendo:

- w: Ancho de celdas. Si 0, estos se extienden hasta el margen derecho de la página.
- H : Alto de las celdas.
- Txt:Cadena para imprimir.
- Border : Indica si los bordes deben ser dibujados al rededor del bloque de la celda. El valor puede ser un número:
    - 0: no borde
    - 1: marco
        - algún o todos los siguientes caracteres:
    - L: izquierda
    - T: superior
    - R: derecha
    - B: inferior
    - Su valor por defecto es 0.
- align : Establece la alineación de texto.
    - L: a la izquierda
    - C: centrado
    - R: a la derecha
    - J: justificación (valor por defecto)
- fill : Indica si el fondo de la celda debe ser dibujado (true) o transparente (false). Valor por defecto: false.

9.2.6 Tablas en FPDF

Vamos a ver dos ejemplos de construcción de tablas, uno más sencillo que el otro, pero ambos igual de interesantes.

El primer ejemplo es un ejemplo básico, tan solo 2 lineas y la cabecera. Para ello vamos a pasarle un array con los nombres de cada celda, y después vamos a ir rellenando las columnas y las filas con la función CELL.

La función seria la siguiente:

```
functionTablaBasica($header)
   {
    //Cabecera
    foreach($header as $col)
    $this->Cell(40,7,$col,1);
    $this->Ln();
```

```
      $this->Cell(40,5,"hola",1);
      $this->Cell(40,5,"hola2",1);
      $this->Cell(40,5,"hola3",1);
      $this->Cell(40,5,"hola4",1);
      $this->Ln();
      $this->Cell(40,5,"linea ",1);
      $this->Cell(40,5,"linea 2",1);
      $this->Cell(40,5,"linea 3",1);
      $this->Cell(40,5,"linea 4",1);
   }
```

y la llamada a la función seria la siguiente:

```
//Creación del objeto de la clase heredada
$pdf=new PDF();
//Títulos de las columnas
$header=array('Columna 1','Columna 2','Columna 3','Columna 4');
$pdf->AliasNbPages();
//Primera página
$pdf->AddPage();
$pdf->SetY(65);
//$pdf->AddPage();
$pdf->Tabla Basica($header);
```

El siguiente ejemplo de tabla es algo más complicado ya que le vamos a dar colores a las filas y vamos a quitar los bordes inferiores de cala linea.

La función sería la siguiente:

```
function TablaColores($header)
{
//Colores, ancho de línea y fuente en negrita
$this->SetFillColor(255,0,0);
$this->SetTextColor(255);
$this->SetDrawColor(128,0,0);
$this->SetLineWidth(.3);
$this->SetFont('','B');
//Cabecera

for($i=0;$i<count($header);$i++)
$this->Cell(40,7,$header[$i],1,0,'C',1);
$this->Ln();
//Restauración de colores y fuentes
$this->SetFillColor(224,235,255);
$this->SetTextColor(0);
$this->SetFont('');
//Datos
$fill=false;

$this->Cell(40,6,"hola",'LR',0,'L',$fill);
$this->Cell(40,6,"hola2",'LR',0,'L',$fill);
$this->Cell(40,6,"hola3",'LR',0,'R',$fill);
$this->Cell(40,6,"hola4",'LR',0,'R',$fill);
$this->Ln();
$fill=true;
$this->Cell(40,6,"col",'LR',0,'L',$fill);
$this->Cell(40,6,"col2",'LR',0,'L',$fill);
$this->Cell(40,6,"col3",'LR',0,'R',$fill);
$this->Cell(40,6,"col4",'LR',0,'R',$fill);
$fill=!$fill;
$this->Ln();
```

```
$this->Cell(160,0,'','T');
}
```

Hemos establecido el color de relleno de las celdas, el color del texto, el grosor de los bordes y el tipo de letra. A continuación, hemos creado la primera linea en rojo y la letra en negrita. Después asignamos otros colores para el relleno y la letra y vamos rellenando el resto de las filas. Vamos colocando los bordes donde necesitamos y diciendo si la celda va rellena o no de color con la variable $fill.

Al final tenemos que añadir el borde de abajo del todo, para ello utilizamos la linea:

```
$this->Cell(160,0,'','T');
```

Que nos dice que la linea tiene que ser 160 de ancho, 0 de alto y mostrando solo el borde superior.