PdfPrinter

La clase THaruPdf intenta proveer las mismas funciones de la clase TPrinter de FW de modo que se pueda usar en su reemplazo con el mínimo de cambios.

**Declaración del objeto de impresión**

Donde antes escribíamos

DEFINE PRINTER oPrint ...

ahora escribimos

oPrint:= THaruPdf():New()

opcionalmente podemos indicar desde el principio el nombre del pdf y los valores de las contraseńas. Los parámetros completos son

oPrint:= THaruPdf():New( cFileName, cPassword, cOwnerPassword, nPermission )

cPassword es la clave para acceder al pdf con los limites indicados por nPermission, cOwnerPassword es para poder modificarlo. nPermission se establece por defecto como HPDF\_ENABLE\_READ + HPDF\_ENABLE\_PRINT + HPDF\_ENABLE\_COPY Los parámetros de seguridad son totalmente opcionales.

Si estos valores se establecen al crear el objeto, cuando se invoque a End() automáticamente grabará el pdf. Si no se indica nada al crear el objeto, se graba manualmente con el método Save

oPrint:Save( cNombreDelPdf ) // --> devuelve un valor numérico

El método Save() devuelve un valor numérico. Si todo está bien, devuelve 0, un valor diferente de 0 indica algún error de Haru.

**Inicio y Fin de página**

La separación de páginas se hace de forma usual:

oPrint:StartPage()

oPrint:EndPage()

**Fuentes**

La declaración de fuentes es diferente. Con FW se escribía

DEFINE FONT oFont1 NAME "COURIER NEW" SIZE 0, -8 Of oPrint

con THaruPdf :

oFont1:= oPrint:DefineFont( 'Courier', 8 )

Un detalle importante es que tipo de fuente utilizamos, si es una predefinida o una fuente TrueType.

Las fuentes **predefinidas** (conocidas como Base14) son las más livianas en términos del tamańo del pdf generado, porque no se insertan en el pdf al estar ya incluídas en los lectores de pdf como el Acrobat Reader, Foxit, etc. Son las siguientes: Courier Courier-Bold Courier-Oblique Courier-BoldOblique Helvetica Helvetica-Bold Helvetica-Oblique Helvetica-BoldOblique Times-Roman Times-Bold Times-Italic Times-BoldItalic Symbol ZapfDingbats

Las fuentes **TrueType** deben ser incluídas en el pdf generado, por lo tanto el fichero <fuente>.ttf debe estar disponible. El nombre de las fuentes no coincide, en general, con el nombre del fichero, por lo que hay que incluir el nombre del fichero en la declaración de las fuentes usando la función:

HaruAddFont( cFontName, cTtfFile )

*cFontName* es el nombre de la fuente, y *cTtfFile* el fichero .ttf que lo define. Por comodidad y frecuencia de uso, ya están predefinidas las siguientes fuentes, por lo que no es necesario declararlas.

Arial

Verdana

Courier New

Calibri

Tahoma

Tambien, por comodidad, la función HaruAddFont busca el fichero en el directorio actual, y si no lo encuentra, en el directorio de fuentes de Windows. Podemos declarar un directorio alternativo para incluir nuestras propias fuentes con la función:

SetHaruFontDir(cDir)

donde cDir es el directorio donde queremos que busque las fuentes.

**Imagenes**

De momento solo soporta imágenes en formato PNG y JPG. Atención/Achtung/Warning: Las pruebas realizadas indican que usar pngs generados con Paint producen pdfs enormes y son muy lentos. La librería ha sido optimizada para cargar las imágenes de forma indexada, es decir que cuando repetimos una imagen en varias páginas, la librería carga la imagen dentro del pdf una sola vez, y usa la misma copia de imagen cada vez que se referencia, por ejemplo como fondo de cada página.

**Como guardar el fichero**

No hace falta indicar ENDPRINT ni destruir las fuentes, es decir ENDPRINT no hace falta oFont:End() no hace falta

**Poner contraseńa.**

La contraseńa del usuario y/o propietario se pueden indicar en el momento de creación del objeto THaruPdf, o bien asignar manualmente las DATAs de la clase antes de llamar al método Save() o End()

DATA nPermission

DATA cPassword

DATA cOwnerPassword

**Como enlazar**

Además de incluir *PdfPrinter.lib*, se requiere *libhpdf.lib* y *png.lib*, que se encuentran en el mismo directorio del proyecto de la librería, y las librerías *hbhpdf.lib* y *hbzlib.lib* del propio Harbour.

**Límites**

El límite más importante es el de la cantidad de páginas, que es bastante alto. Generando nóminas con un bitmap de fondo se han generado pdfs de hasta 8000 páginas. Aparentemente superando esa cantidad de páginas se producen errores por la falta de capacidad de las estructuras internas de los pdfs, que requiere una reestructuración interna de la paginación que Haru no es capaz de hacer.

**Errores, cosas que faltan...**

Como es una versión muy temprana, es probable que algo no funcione como se espera o echemos en falta alguna funcionalidad. Para poder implementar nuevas funciones o corregir/modificar las existentes, la clase está montada con el patrón **'variaciones protegidas'**, usando como fachada una clase intermedia vacía (THaruPDF), que hereda toda la funcionalidad de la implementación real (THaruPDFBase). Esto tiene como objeto permitirnos introducir ajustes y nuevas funciones sin tener que modificar el código original. Si queremos modificar algo de la clase, EN NUESTRO PROYECTO crearemos un nuevo prg y declararemos la clase THaruPDF que herede de THaruPDFBase (podemos copiar el prg vacio que ya está en el proyecto de la librería), y ańadiremos allí todo el nuevo código. Con esto lograremos que la clase haga lo que queremos o ańadamos alguna prestación sin interferir con el código público de la librería, usado en todos los proyectos.

#include 'hbclass.ch'

#include 'harupdf.ch'

#define \_\_NODEBUG\_\_

#include 'debug.ch'

//-------------------------------------------------------------------------

CLASS THaruPDF FROM THaruPDFBase

// Clase intermedia para proveer de variaciones protegidas

ENDCLASS

Cuando alguna nueva funcionalidad se incorpore a la librería, automáticamente se incluirá en nuestro proyecto, sin afectar la parte local del código adicional que ya existe.

**Ejemplo de uso**

En la carpeta \PrinterPdfTest hay un proyecto con el ejemplo de uso para imprimir una nómina. Incluye algunos ejemplos complejos de uso, como la adaptación de un texto a un recuadro.

El código está a continuación.

/\*

\* Proyecto: PdfPrinterTest

\* Fichero: PdfPrinterTest.prg

\* Descripción:

\* Autor:

\* Fecha: 23/02/2015

Ejemplo de Impresión de una nómina usando librería PdfPrinter

\*/

#define CRLF HB\_OSNewLine()

// Definicion segun la clase TPrinter de FW

#define PAD\_LEFT 0

#define PAD\_RIGHT 1

#define PAD\_CENTER 2

// Definiciones de alineacion segun la libreria Haru

#define HPDF\_TALIGN\_LEFT 0

#define HPDF\_TALIGN\_RIGHT 1

#define HPDF\_TALIGN\_CENTER 2

#define HPDF\_TALIGN\_JUSTIFY 3

//------------------------------------------------------------------------------

FUNCTION Main()

//------------------------------------------------------------------------------

LOCAL oPrint, aSizes, j, oFont, oFont1, oFont2, nLinea

SetMode( 30, 80 )

// Creamos el objeto de impresión

oPrint:= THaruPdf():New()

// oPrint:SetCompression( HPDF\_COMP\_ALL )

// Definimos las fuentes que se van a usar en el pdf

// DEFINE FONT oFont1 NAME "Arial" SIZE 0, -8 Of oPrint

// DEFINE FONT oFont2 NAME "COURIER NEW" SIZE 0, -6 Of oPrint

oFont1:= oPrint:DefineFont( 'Arial', 8)

oFont2:= oPrint:DefineFont( 'Courier', 6 )

oPrint:StartPage()

//Imprimir imagen

aSizes:= oPrint:SizeInch2Pix( 29.7/2.54, 21.0/2.54 )

oPrint:SAYBITMAP(0, 0, "Nomina2.png", aSizes[1], aSizes[2] )

// Rellenar los campos de texto

// Datos de la empresa

oPrint:CMSAY(4.00, 4.70, '07544', oFont1, , , , 0)

oPrint:CMSAY(4.50, 4.70, 'Endesa Generación S.A.', oFont2, , 0xff0000, , 0)

oPrint:CMSAY(5.10, 4.70, 'A80233245', oFont1, , , , 0)

oPrint:CMSAY(5.60, 4.70, '23992001J', oFont1, , , , 0)

oPrint:CMSAY(6.20, 4.70, 'Abril 2014', oFont1, , , , 0)

//Imprimir cuadro de info de cliente

oPrint:CMSAY(3.80, 10.40, 'Mortadelo y Filemón', oFont1, , , , 0)

// Cuerpo del recibo

// Como prueba alternamos la fuente

nLinea:=7.80

FOR j:= 1 TO 10

IF j % 2 == 0

oFont:= oFont1

ELSE

oFont:= oFont2

ENDIF

oPrint:CMSAY(nLinea, 1.60, alltrim('field->concepto'), oFont, , , , 0)

oPrint:CMSAY(nLinea, 12.10, alltrim(formatearNumero(20)), oFont, , , , PAD\_RIGHT)

oPrint:CMSAY(nLinea, 14.00, alltrim(formatearNumero(21.33)), oFont, , , , PAD\_RIGHT)

oPrint:CMSAY(nLinea, 15.90, alltrim(formatearNumero(312321)), oFont, , , , PAD\_RIGHT)

oPrint:CMSAY(nLinea, 17.90, alltrim(formatearNumero(3123.333)), oFont, , , , PAD\_RIGHT)

oPrint:CMSAY(nLinea, 19.80, alltrim(formatearNumero(132312)), oFont, , , , PAD\_RIGHT)

nLinea+= 0.5

NEXT

//Imprimir cuadros CC y líquido a percibir

oPrint:CMSAY(15.40, 15.90, alltrim(formatearNumero(132321)), oFont1, , , , PAD\_RIGHT)

oPrint:CMSAY(15.40, 17.90, alltrim(formatearNumero(132132)), oFont1, , , , PAD\_RIGHT)

oPrint:CMSAY(15.40, 19.80, alltrim(formatearNumero(312123)), oFont1, , , , PAD\_RIGHT)

oPrint:CMSAY(16.10, 10.90, alltrim('231321'), oFont1, , , , PAD\_RIGHT)

oPrint:CMSAY(16.35, 19.70, alltrim(formatearNumero(213.2)), oFont1, , , , PAD\_RIGHT)

oPrint:CMSAY(16.50, 10.90, alltrim('1232-3121-33-998877667721'), oFont1, , , , PAD\_RIGHT)

//Imprimir informativos

nlinea:=18.30

ncolumna1:=1.60

ncolumna2:=10.50

for j:= 1 to 4

oPrint:CMSAY(nlinea, ncolumna1, 'SUBSTR(field->concepto,1,30)', oFont1, , , , 0)

oPrint:CMSAY(nlinea, ncolumna2, Str(8923.22, 15, 2), oFont1, , , , PAD\_RIGHT)

nlinea+= 0.40

if nlinea >= 20.65

nlinea:=18.30

ncolumna1:= 10.70

ncolumna2:= 19.80

endif

end

//Imprimir cuadro subsidio

cTexto:= "Si es Vd. perceptor del subsidio por desempleo, le recordamos que debe "+;

"acudir a la oficina del INEM los meses en que la suma de su patrimonio "+;

"(alquileres de arrendamientos urbanos, intereses bancarios mensuales, herencias, "+;

"etc..) más rentas diferentes al complemento empresa (ayuda escolar, imputación "+;

"fiscal de fluido eléctrico, de Seguro de vida, de Póliza de salud y la aportación "+;

"por parte de la empresa al plan de pensiones, entre otras..) supera el 75 % del "+;

"S.M.I mensual, para declarar dicha situación ante el SPEE."+CRLF +;

CRLF +;

"En caso de que hubiese comunicación de suspensión o extinción, a la mayor "+;

"brevedad posible, deben comunicarlo a la Entidad Gestora Minera, por las vías "+;

"que tienen establecidas."

//cTexto:= ajustarTexto(cTexto)

aTexto := ajustarTexto(oPrint,oFont2,upper(cTexto), 510) //3250

nLongarray:= len(aTexto)

nLinea:= 24.10

If nLongarray >= 1

For j:= 1 to nLongarray

oPrint:CMSAY(nLinea, 1.60, aTexto[j], oFont2, 510, , , 0)

If nLongarray != 0 .and. j!= nLongarray

nLinea+=0.30

EndIf

Next

Endif

oPrint:CMSAY(28.10, 19.60, alltrim('pasivos->correos'), oFont2, , , , PAD\_RIGHT)

oPrint:EndPage()

oPrint:Save( 'Nomina1.pdf' )

RETURN NIL

//-----------------------------------------------------------------------STATIC FUNCTION FormatearNumero(nValor)

//-----------------------------------------------------------------------

RETURN LTrim(Transform(nValor, '@E 999,999,999.99'))

//-----------------------------------------------------------------------STATIC FUNCTION AjustarTexto(oPrint, oFont, cTexto, nPixeles)

//----------------------------------------------------------------------- LOCAL aSalida:={}, aLinea:={}, part, i:=1

cTexto := alltrim(cTexto)

IF at(CRLF, cTexto)!=0

WHILE at(CRLF, cTexto)!=0

aAdd(aLinea, substr(cTexto, 1, at(CRLF, cTexto)-1))

cTexto := substr(cTexto, at(CRLF, cTexto)+1, len(cTexto))

ENDDO

ENDIF

aAdd(aLinea, cTexto)

FOR i:=1 to len(aLinea)

IF oPrint:GetTextWidth(aLinea[i], oFont) > nPixeles

WHILE oPrint:GetTextWidth(aLinea[i], oFont) > nPixeles

Part:=alltrim(Substr(aLinea[i], 1, rat(" ", aLinea[i])))

WHILE oPrint:GetTextWidth(Part, oFont) > nPixeles

Part:=alltrim(Substr(part, 1, rat(" ", Part)))

END

aadd(aSalida, Part)

aLinea[i]:=alltrim(Substr(aLinea[i], len(part)+2))

part:=aLinea[i]

END

IF ascan(aSalida, Part) == 0 .and. !empty(part)

aadd(aSalida, Part)

ENDIF

ELSE

aadd(aSalida, aLinea[i])

ENDIF

NEXT

RETURN aSalida