using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.IO;

using iTextSharp.text;

using iTextSharp.text.pdf;

using System.Data;

namespace DoNetPdf

{

/// <summary>

/// PDF打印功能实现类

/// </summary>

public class PdfPrinter : PdfPageEventHelper,IPdfPageEvent

{

/// <summary>

/// 默认字体：兼容亚洲文字

/// </summary>

public Font DefaultFont { get; set; }

/// <summary>

/// PDF文件导出路径

/// </summary>

private string PrintPath { get; set; }

/// <summary>

/// 默认纸张大小

/// </summary>

public Rectangle DefaultPageSize { get; set; }

/// <summary>

/// 页眉内容集合

/// </summary>

public PdfHeaderContent HeaderContent { get; set; }

/// <summary>

/// 全局PDF文档对象

/// </summary>

private Document Doc { get; set; }

/// <summary>

/// 构造函数

/// </summary>

/// <param name="printPath">指定PDF文件导出路径</param>

public PdfPrinter(string printPath)

{

DefaultFont = new Font(BaseFont.CreateFont(@"C:\Windows\Fonts\meiryo.ttc,0", BaseFont.IDENTITY\_H, BaseFont.NOT\_EMBEDDED));

PrintPath = printPath;

DefaultPageSize = PageSize.A4;

}

/// <summary>

/// 构造函数

/// </summary>

/// <param name="printPath">指定PDF文件导出路径</param>

/// <param name="paperSize">纸张大小</param>

/// <param name="headerContent">PDF文件页眉内容</param>

public PdfPrinter(string printPath, Rectangle paperSize, PdfHeaderContent headerContent)

{

DefaultFont = new Font(BaseFont.CreateFont(@"C:\Windows\Fonts\meiryo.ttc,0", BaseFont.IDENTITY\_H, BaseFont.NOT\_EMBEDDED));

PrintPath = printPath;

DefaultPageSize = paperSize;

HeaderContent = headerContent;

}

/// <summary>

/// 初始化文档

/// </summary>

public void Init()

{

if (Doc == null)

{

Doc = new Document(DefaultPageSize, 40f, 40f, 83f, 20f);

FileStream fs = new FileStream(PrintPath, FileMode.Create);

PdfWriter writer = PdfWriter.GetInstance(Doc, fs);

if (HeaderContent != null)

writer.PageEvent = new PdfPrinter(this.PrintPath, this.DefaultPageSize, this.HeaderContent);

}

if (!Doc.IsOpen())

{

//创建文件

Doc.Open();

}

}

/// <summary>

/// 保存并释放文档

/// </summary>

public void Close()

{

Doc.Close();

}

/// <summary>

/// 绘制页眉

/// </summary>

/// <param name="writer"></param>

/// <param name="document"></param>

public override void OnStartPage(PdfWriter writer, Document document)

{

float strWidth = 0f;

float fontSize = 0f;

float padTop = 40f;

PdfContentByte cb = writer.DirectContent;

cb.BeginText();

//标题

if (!string.IsNullOrEmpty(HeaderContent.HeaderTitle))

{

fontSize = 14f;

strWidth = HeaderContent.HeaderTitle.Length \* fontSize;

cb.SetFontAndSize(DefaultFont.BaseFont, fontSize);

cb.SetCharacterSpacing(3);

cb.SetTextMatrix((document.PageSize.Width - strWidth) / 2, document.PageSize.Height - padTop);

cb.ShowText(HeaderContent.HeaderTitle);

}

//时间、页码

string str = DateTime.Now.ToString("yyyy年MM月dd日 HH:mm") + " 第" + writer.PageNumber + "頁";

fontSize = 10f;

strWidth = str.Length \* fontSize;

cb.SetFontAndSize(DefaultFont.BaseFont, fontSize);

cb.SetCharacterSpacing(2);

cb.SetTextMatrix(document.PageSize.Width - strWidth, document.PageSize.Height - padTop + 2);

cb.ShowText(str);

//标题列A

padTop = 55f;

if (HeaderContent.HeaderNamesA != null && HeaderContent.HeaderNamesA.Count > 0)

{

List<float> positions = HeaderContent.HeaderPositionsA;

bool isPosEmpty = (positions == null || positions.Count != HeaderContent.HeaderNamesA.Count);//坐标为空或与表头数量不一致时默认位置

fontSize = 12f;

cb.SetFontAndSize(DefaultFont.BaseFont, fontSize);

cb.SetCharacterSpacing(0);

float left = document.LeftMargin;//默认起始位置

for (int i = 0; i < HeaderContent.HeaderNamesA.Count; i++)

{

if(!isPosEmpty)//如果指定了详细位置则按照指定位置绘制否则自动计算

left = HeaderContent.HeaderPositionsA[i];

cb.SetTextMatrix(left, document.PageSize.Height - padTop);

str = HeaderContent.HeaderNamesA[i];

cb.ShowText(str);

//left += str.Length \* fontSize + 20f;

float step = (document.PageSize.Width - document.LeftMargin \* 2) / HeaderContent.HeaderNamesA.Count;

left += step;

}

}

//标题列B

padTop = 68f;

if (HeaderContent.HeaderNamesB != null && HeaderContent.HeaderNamesB.Count > 0)

{

List<float> positions = HeaderContent.HeaderPositionsB;

bool isPosEmpty = (positions == null || positions.Count != HeaderContent.HeaderNamesB.Count);//坐标为空或与表头数量不一致时默认位置

fontSize = 12f;

cb.SetFontAndSize(DefaultFont.BaseFont, fontSize);

cb.SetCharacterSpacing(0);

float left = document.LeftMargin;//默认起始位置

for (int i = 0; i < HeaderContent.HeaderNamesB.Count; i++)

{

if (!isPosEmpty)//如果指定了详细位置则按照指定位置绘制否则自动计算

left = HeaderContent.HeaderPositionsB[i];

cb.SetTextMatrix(left, document.PageSize.Height - padTop);

str = HeaderContent.HeaderNamesB[i];

cb.ShowText(str);

//left += str.Length \* fontSize + 20f;

float step = (document.PageSize.Width - document.LeftMargin \* 2) / HeaderContent.HeaderNamesA.Count;

left += step;

}

}

//标题列C

padTop = 80f;

if (HeaderContent.HeaderNamesC != null && HeaderContent.HeaderNamesC.Count > 0)

{

List<float> positions = HeaderContent.HeaderPositionsC;

bool isPosEmpty = (positions == null || positions.Count != HeaderContent.HeaderNamesC.Count);//坐标为空或与表头数量不一致时默认位置

fontSize = 12f;

cb.SetFontAndSize(DefaultFont.BaseFont, fontSize);

cb.SetCharacterSpacing(0);

float left = document.LeftMargin;//默认起始位置

for (int i = 0; i < HeaderContent.HeaderNamesC.Count; i++)

{

if (!isPosEmpty)//如果指定了详细位置则按照指定位置绘制否则自动计算

left = HeaderContent.HeaderPositionsC[i];

cb.SetTextMatrix(left, document.PageSize.Height - padTop);

str = HeaderContent.HeaderNamesC[i];

cb.ShowText(str);

//left += str.Length \* fontSize + 20f;

float step = (document.PageSize.Width - document.LeftMargin \* 2) / HeaderContent.HeaderNamesA.Count;

left += step;

}

}

cb.EndText();

//绘制分割线

padTop = 83f;

cb.MoveTo(document.LeftMargin,document.PageSize.Height-padTop);

cb.LineTo(document.PageSize.Width - document.RightMargin, document.PageSize.Height - padTop);

cb.SetLineDash(3f,3f);

cb.Stroke();

base.OnStartPage(writer, document);

}

/// <summary>

/// 打印一段文字

/// </summary>

/// <param name="text">文字内容</param>

/// <param name="align">文字对齐 0左、1中、2右</param>

public void PrintString(string text, int align)

{

Paragraph ps = new Paragraph(text, DefaultFont);

ps.Alignment = align;

Doc.Add(ps);

}

/// <summary>

/// 打印一行表格数据

/// </summary>

/// <param name="pCells">数据行包含的单元格集合</param>

/// <param name="cWidth">数据行包含的单元格各自的宽度集合</param>

public void PrintTableRow(PdfPCell[] pCells, float[] cWidth)

{

if (pCells == null || pCells.Length <= 0)

throw new ArgumentNullException("can not found any cell.");

PdfPTable pt = new PdfPTable(pCells.Length);

pt.WidthPercentage = 100f;

pt.SkipFirstHeader = false;

PdfPRow pRow = new PdfPRow(pCells);

pt.Rows.Add(pRow);

if (cWidth != null && cWidth.Length == pCells.Length)

pt.SetWidths(cWidth);

Doc.Add(pt);

}

/// <summary>

/// 打印一个表格数据

/// </summary>

/// <param name="table">表格</param>

/// <param name="cWidth">宽度</param>

/// <param name="cAlign">对齐 0左 1中 2右</param>

public void PrintTableContent(DataTable table, float[] cWidth, int[] cAlign)

{

if (table == null || table.Columns == null || table.Columns.Count <= 0)

throw new ArgumentNullException("can not found any column from table.");

PdfPTable pt = new PdfPTable(table.Columns.Count);

pt.WidthPercentage = 100f;

pt.SkipFirstHeader = false;

PdfPCell[] pCells = new PdfPCell[table.Columns.Count];

PdfPRow pRow = null;

//内容

foreach (DataRow row in table.Rows)

{

pCells = new PdfPCell[table.Columns.Count];

for (int i = 0; i < table.Columns.Count; i++)

{

pCells[i] = new PdfPCell(new Phrase(row[i].ToString(), DefaultFont));

if (cAlign != null && cAlign.Length == table.Columns.Count)

{

pCells[i].HorizontalAlignment = cAlign[i];

pCells[i].Border = 0;

}

pCells[i].VerticalAlignment = 1;

}

pRow = new PdfPRow(pCells);

pt.Rows.Add(pRow);

}

if (cWidth != null && cWidth.Length == table.Columns.Count)

pt.SetWidths(cWidth);

Doc.Add(pt);

}

/// <summary>

/// 换一页

/// </summary>

public void NewPage()

{

Doc.NewPage();

}

}

/// <summary>

/// PDF打印页眉内容集合类

/// </summary>

public class PdfHeaderContent

{

/// <summary>

/// 构造函数

/// </summary>

/// <param name="title">标题文字</param>

public PdfHeaderContent(string title)

{

HeaderTitle = title;

}

/// <summary>

/// 标题文字

/// </summary>

public string HeaderTitle

{

get;

set;

}

/// <summary>

/// 表头集合A

/// </summary>

public List<string> HeaderNamesA

{

get;

set;

}

/// <summary>

/// 表头坐标集合A

/// </summary>

public List<float> HeaderPositionsA

{

get;

set;

}

/// <summary>

/// 表头集合B

/// </summary>

public List<string> HeaderNamesB

{

get;

set;

}

/// <summary>

/// 表头坐标集合B

/// </summary>

public List<float> HeaderPositionsB

{

get;

set;

}

/// <summary>

/// 表头集合C

/// </summary>

public List<string> HeaderNamesC

{

get;

set;

}

/// <summary>

/// 表头坐标集合C

/// </summary>

public List<float> HeaderPositionsC

{

get;

set;

}

}

}