# XDoc.PDF Developer Guide – Document Creator Module

## Table of Contents

# How to create a file

## Create a “Hello World” file

```
String outputFilePath = Program.RootPath + "\\" + "Sample1.pdf";

Document document = new Document();
PDFBuildHandler.Create(document, outputFilePath);

//  open document
document.Open();

//  add content to the document
document.Add(new Paragraph("Hello World"));

//  close document and save to the output file
document.Close();
```

## Add a new page to the document

```
String outputFilePath = Program.RootPath + "\\" + "Sample2.pdf";

Document document = new Document();
PDFBuildHandler.Create(document, outputFilePath);

//  open document
document.Open();

document.Add(new Paragraph("This is the 1st page."));
//  add a new page and move current position to the next page
document.NewPage();
document.Add(new Paragraph("This is the 2nd page."));
//  add a new page and move current position to the next page
document.NewPage();

//  close document and save to the output file
document.Close();
```

## Create a file with paragraphs

```
String outputFilePath = Program.RootPath + "\\" + "Sample3.pdf";

String[] contents = new String[] {
    "Charles Babbage, an English mechanical engineer and polymath, originated the concept of a programmable computer. Considered the \"father of the computer\", he conceptualized and invented the first mechanical computer in the early 19th century. After working on his revolutionary difference engine, designed to aid in navigational calculations, in 1833 he realized that a much more general design, an Analytical Engine, was possible. The input of programs and data was to be provided to the machine via punched cards, a method being used at the time to direct mechanical looms such as the Jacquard loom. For output, the machine would have a printer, a curve plotter and a bell. The machine would also be able to punch numbers onto cards to be read in later. The Engine incorporated an arithmetic logic unit, control flow in the form of conditional branching and loops, and integrated memory, making it the first design for a general-purpose computer that could be described in modern terms as Turing-complete.",
    "The machine was about a century ahead of its time. All the parts for his machine had to be made by hand ——this was a major problem for a device with thousands of parts. Eventually, the project was dissolved with the decision of the British Government to cease funding. Babbage's failure to complete the analytical engine can be chiefly attributed to difficulties not only of politics and financing, but also to his desire to develop an increasingly sophisticated computer and to move ahead faster than anyone else could follow. Nevertheless, his son, Henry Babbage, completed a simplified version of the analytical engine's computing unit (the mill) in 1888. He gave a successful demonstration of its use in computing tables in 1906.",
    "The principle of the modern computer was proposed by Alan Turing, in his seminal 1936 paper. Turing proposed a simple device that he called \"Universal Computing machine\" that is later known as a Universal Turing machine. He proved that such machine is capable of computing anything that is computable by executing instructions (program) stored on tape, allowing the machine to be programmable. The fundamental concept of Turing's design is stored program, where all instruction for computing is stored in the memory. Von Neumann acknowledged that the central concept of the modern computer was due to this paper. Turing machines are to this day a central object of study in theory of computation. Except for the limitations imposed by their finite memory stores, modern computers are said to be Turing-complete, which is to say, they have algorithm execution capability equivalent to a universal Turing machine.",
    "Early computing machines had fixed programs. Changing its function required the re-wiring and re-structuring of the machine. With the proposal of the stored-program computer this changed. A stored-program computer includes by design an instruction set and can store in memory a set of instructions (a program) that details the computation. The theoretical basis for the stored-program computer was laid by Alan Turing in his 1936 paper. In 1945 Turing joined the National Physical Laboratory and began work on developing an electronic stored-program digital computer. His 1945 report 'Proposed Electronic Calculator'was the first specification for such a device. John von Neumann at the University of Pennsylvania also circulated his First Draft of a Report on the EDVAC in 1945.",
    "The Manchester Small-Scale Experimental Machine, nicknamed Baby, was the world's first stored-program computer. It was built at the Victoria University of Manchester by Frederic C. Williams, Tom Kilburn and Geoff Tootill, and ran its first program on 21 June 1948. It was designed as a testbed for the Williams tube the first random-access digital storage device. Although the computer was considered \"small and primitive\" by the standards of its time, it was the first working machine to contain all of the elements essential to a modern electronic computer. As soon as the SSEM had demonstrated the feasibility of its design, a project was initiated at the university to develop it into a more usable computer, the Manchester Mark 1."
```

```
};

Document document = new Document();
PDFBuildHandler.Create(document, outputFilePath);

//  open document
document.Open();

//  add content to the document
foreach (String content in contents)
{
    document.Add(new Paragraph(content));
}

//  close document and save to the output file
document.Close();
```

## Set page size for a document

```
String outputFilePath = Program.RootPath + "\\" + "Sample4.pdf";

Document document = new Document();
PDFBuildHandler.Create(document, outputFilePath);

document.SetPageSize(PaperSize.A4);

//  open document
document.Open();

document.Add(new Paragraph("Page Size: A4"));

document.SetPageSize(PaperSize.Letter);
document.NewPage();
document.Add(new Paragraph("Page Size: Letter"));

document.SetPageSize(4F, 3F);
document.NewPage();
document.Add(new Paragraph("Page Size: Width = 4 inches; Height = 3 inches"));

//  close document and save to the output file
document.Close();
```

## Set page margins for a document

```
String outputFilePath = Program.RootPath + "\\" + "Sample5.pdf";

Document document = new Document();
PDFBuildHandler.Create(document, outputFilePath);

float leftMargin = 0F;
float rightMargin = 0F;
float topMargin = 0F;
float bottomMargin = 0F;
document.SetMargins(leftMargin, rightMargin, topMargin, bottomMargin);

//  open document
document.Open();

//  add content to the document
document.Add(new Paragraph("Left Margin = 0; Right Margin = 0; Top Margin = 0; Bottom Margin =
0"));

leftMargin = 144F;
rightMargin = 144F;
topMargin = 144F;
bottomMargin = 144F;
document.SetMargins(leftMargin, rightMargin, topMargin, bottomMargin);
document.NewPage();

//  add content to the document
document.Add(new Paragraph("Left Margin = 144 pt. (2 inches); Right Margin = 144 pt. (2 inches);
Top Margin = 144 pt. (2 inches); Bottom Margin = 144 pt. (2 inches)"));

//  close document and save to the output file
document.Close();
```

# How to add a paragraph

## Set paragraph indentation

```
String outputFilePath = Program.RootPath + "\\" + "Sample6.pdf";

Document document = new Document();
PDFBuildHandler.Create(document, outputFilePath);

//  open document
document.Open();

Paragraph p1 = new Paragraph("This is a sample paragraph that the first line indentation is 0
pt., the left indentation is 0 pt.and the right indentation it 0 pt.");
p1.Alignment = Alignment.ALIGN_JUSTIFIED;
p1.FirstLineIndent = 0F;
p1.IndentationLeft = 0F;
p1.IndentationRight = 0F;
document.Add(p1);

Paragraph p2 = new Paragraph("This is a sample paragraph that the first line indentation is -20
pt., the left indentation is 50 pt.and the right indentation it 50 pt.");
p2.Alignment = Alignment.ALIGN_JUSTIFIED;
p2.FirstLineIndent = -20F;
p2.IndentationLeft = 50F;
p2.IndentationRight = 50F;
document.Add(p2);

Paragraph p3 = new Paragraph("This is a sample paragraph that the first line indentation is 20
pt., the left indentation is 50 pt.and the right indentation it 50 pt.");
p3.Alignment = Alignment.ALIGN_JUSTIFIED;
p3.FirstLineIndent = 20F;
p3.IndentationLeft = 50F;
p3.IndentationRight = 50F;
document.Add(p3);

//  close document and save to the output file
document.Close();
```

## Set paragraph before and after space

```
String outputFilePath = Program.RootPath + "\\" + "Sample7.pdf";

Document document = new Document();
PDFBuildHandler.Create(document, outputFilePath);

//  open document
document.Open();

Paragraph p1 = new Paragraph("This is a sample paragraph that the before paragraph spacing is 0 pt. and the after paragraph spacing is 0 pt.");
p1.SpacingBefore = 0;
p1.SpacingAfter = 0;
document.Add(p1);

Paragraph p2 = new Paragraph("This is a sample paragraph that the before paragraph spacing is 20 pt. and the after paragraph spacing is 100 pt.");
p2.SpacingBefore = 20;
p2.SpacingAfter = 100;
document.Add(p2);

Paragraph p3 = new Paragraph("This is a sample paragraph that the before paragraph spacing is 0 pt. and the after paragraph spacing is 0 pt.");
p3.SpacingBefore = 0;
p3.SpacingAfter = 0;
document.Add(p3);

//  close document and save to the output file
document.Close();
```

## Set line space in the paragraph

```
String outputFilePath = Program.RootPath + "\\" + "Sample8.pdf";
String content = "The Solar System is the gravitationally bound system comprising the Sun and
the objects that orbit it, either directly or indirectly. Of those objects that orbit the Sun
directly, the largest eight are the planets, with the remainder being significantly smaller
objects, such as dwarf planets and small Solar System bodies. Of the objects that orbit the Sun
indirectly, the moons, two are larger than the smallest planet, Mercury.";

Document document = new Document();
PDFBuildHandler.Create(document, outputFilePath);

//  open document
document.Open();

Paragraph p1 = new Paragraph(content);
p1.SetLeading(20, 1);
document.Add(p1);

Paragraph p2 = new Paragraph(content);
p2.SetLeading(0, 2);
document.Add(p2);

Paragraph p3 = new Paragraph(content);
p3.SetLeading(50, 2);
document.Add(p3);

//  close document and save to the output file
document.Close();
```

## Set paragraph alignment

```
String outputFilePath = Program.RootPath + "\\" + "Sample9.pdf";
String content = "The Solar System is the gravitationally bound system comprising the Sun and
the objects that orbit it, either directly or indirectly. Of those objects that orbit the Sun
directly, the largest eight are the planets, with the remainder being significantly smaller
objects, such as dwarf planets and small Solar System bodies. Of the objects that orbit the Sun
indirectly, the moons, two are larger than the smallest planet, Mercury.";

Document document = new Document();
PDFBuildHandler.Create(document, outputFilePath);
```

```
//  open document
document.Open();

//  right alignment
Paragraph p1 = new Paragraph(content);
p1.Alignment = Alignment.ALIGN_RIGHT;
document.Add(p1);

//  left alignment
Paragraph p2 = new Paragraph(content);
p2.Alignment = Alignment.ALIGN_LEFT;
document.Add(p2);

//  center alignment
Paragraph p3 = new Paragraph(content);
p3.Alignment = Alignment.ALIGN_CENTER;
document.Add(p3);

//  justified
Paragraph p4 = new Paragraph(content);
p4.Alignment = Alignment.ALIGN_JUSTIFIED;
document.Add(p4);

//  close document and save to the output file
document.Close();
```

## Create a paragraph by the specified font

```
String outputFilePath = Program.RootPath + "\\" + "Sample10.pdf";

Document document = new Document();
PDFBuildHandler.Create(document, outputFilePath);

//  open document
document.Open();

Paragraph p1 = new Paragraph("Font: Helvetica, 36, Regular ", new
Font(Font.FontFamily.Helvetica, 36F, Font.FontStyle.Regular));
document.Add(p1);
Paragraph p2 = new Paragraph("Font: TimesRoman, 36, Regular ", new
Font(Font.FontFamily.TimesRoman, 36F, Font.FontStyle.Regular));
document.Add(p2);
Paragraph p3 = new Paragraph("Font: Courier, 36, Regular ", new Font(Font.FontFamily.Courier,
36F, Font.FontStyle.Regular));
document.Add(p3);
Paragraph p4 = new Paragraph("Font: Helvetica, 12, Bold ", new Font(Font.FontFamily.Helvetica,
12F, Font.FontStyle.Bold));
document.Add(p4);
Paragraph p5 = new Paragraph("Font: Helvetica, 12, Italic ", new Font(Font.FontFamily.Helvetica,
12F, Font.FontStyle.Italic));
```

```
document.Add(p5);

//  close document and save to the output file
document.Close();
```

## Create a paragraph with different fonts

```
String outputFilePath = Program.RootPath + "\\" + "Sample11.pdf";

Document document = new Document();
PDFBuildHandler.Create(document, outputFilePath);

//  open document
document.Open();

Paragraph p = new Paragraph();
p.SetLeading(64, 0);

Chunk c1 = new Chunk("This is the first chunk. ", new Font(Font.FontFamily.Helvetica, 36F,
Font.FontStyle.Regular, new Color(255, 0, 0)));
p.AddChunk(c1);
Chunk c2 = new Chunk("This is the 2nd chunk. ", new Font(Font.FontFamily.Helvetica, 36F,
Font.FontStyle.Italic, new Color(0, 255, 0)));
p.AddChunk(c2);
Chunk c3 = new Chunk("This is the third chunk.", new Font(Font.FontFamily.Helvetica, 18F,
Font.FontStyle.BoldItalic | Font.FontStyle.Underline, new Color(255, 0, 128)));
p.AddChunk(c3);

//  add content to the document
document.Add(p);

//  close document and save to the output file
document.Close();
```

## Apply default underline to text

```
String outputFilePath = Program.RootPath + "\\" + "Sample12.pdf";

Document document = new Document();
```

```
PDFBuildHandler.Create(document, outputFilePath);

//  open document
document.Open();

Paragraph p = new Paragraph();

Font textFont = new Font(Font.FontFamily.Helvetica, 16F, Font.FontStyle.Regular);

Chunk c1 = new Chunk("This is a sample code to apply ", textFont);
p.AddChunk(c1);
Chunk c2 = new Chunk("underline", textFont);
c2.SetUnderline();
p.AddChunk(c2);
Chunk c3 = new Chunk(" to the text.", textFont);
p.AddChunk(c3);

document.Add(p);

//  close document and save to the output file
document.Close();
```

## Apply different underline to text

```
String outputFilePath = Program.RootPath + "\\" + "Sample13.pdf";

Document document = new Document();
PDFBuildHandler.Create(document, outputFilePath);

//  open document
document.Open();

Font textFont = new Font(Font.FontFamily.Helvetica, 16F, Font.FontStyle.Regular);

Chunk c1 = new Chunk("Underline thickness 1 pt; Y position: 0 pt.", textFont);
c1.SetUnderline(1F, 0F);
document.Add(new Paragraph(c1));

Chunk c2 = new Chunk("Underline thickness 1.5 pt; Y position: 5 pt.", textFont);
c2.SetUnderline(1.5F, 5F);
document.Add(new Paragraph(c2));

Chunk c3 = new Chunk("Underline thickness 0.5 pt; Y position: -2 pt.", textFont);
c3.SetUnderline(0.5F, -2F);
document.Add(new Paragraph(c3));

//  close document and save to the output file
document.Close();
```

# How to use Font

## Use standard 14 fonts

```
String outputFilePath = Program.RootPath + "\\" + "Sample12.pdf";

Document document = new Document();
PDFBuildHandler.Create(document, outputFilePath);

//  open document
document.Open();

document.Add(new Paragraph("Font: Helvetica Regular", new Font(Font.FontFamily.Helvetica, 24F,
Font.FontStyle.Regular)));
document.Add(new Paragraph("Font: Helvetica Bold", new Font(Font.FontFamily.Helvetica, 24F,
Font.FontStyle.Bold)));
document.Add(new Paragraph("Font: Helvetica Italic", new Font(Font.FontFamily.Helvetica, 24F,
Font.FontStyle.Italic)));
document.Add(new Paragraph("Font: Helvetica Bold Italic", new Font(Font.FontFamily.Helvetica,
24F, Font.FontStyle.BoldItalic)));

document.Add(new Paragraph("Font: Times New Roman Regular", new Font(Font.FontFamily.TimesRoman,
24F, Font.FontStyle.Regular)));
document.Add(new Paragraph("Font: Times New Roman Bold", new Font(Font.FontFamily.TimesRoman,
24F, Font.FontStyle.Bold)));
document.Add(new Paragraph("Font: Times New Roman Italic", new Font(Font.FontFamily.TimesRoman,
24F, Font.FontStyle.Italic)));
document.Add(new Paragraph("Font: Times New Roman Bold Italic", new
Font(Font.FontFamily.TimesRoman, 24F, Font.FontStyle.BoldItalic)));

document.Add(new Paragraph("Font: Courier Regular", new Font(Font.FontFamily.Courier, 24F,
Font.FontStyle.Regular)));
document.Add(new Paragraph("Font: Courier Bold", new Font(Font.FontFamily.Courier, 24F,
Font.FontStyle.Bold)));
document.Add(new Paragraph("Font: Courier Italic", new Font(Font.FontFamily.Courier, 24F,
Font.FontStyle.Italic)));
document.Add(new Paragraph("Font: Courier Bold Italic", new Font(Font.FontFamily.Courier, 24F,
Font.FontStyle.BoldItalic)));

document.Add(new Paragraph("Font: Symbol", new Font(Font.FontFamily.Symbol, 24F,
Font.FontStyle.Regular)));
document.Add(new Paragraph("Font: ZapfDingbats", new Font(Font.FontFamily.ZapfDingbats, 24F,
Font.FontStyle.Regular)));

//  close document and save to the output file
document.Close();
```

## Create font by importing from a system font

```
String outputFilePath = Program.RootPath + "\\" + "Sample13.pdf";

Document document = new Document();
PDFBuildHandler.Create(document, outputFilePath);

//  open document
document.Open();

Font aFont = Font.CreateFont(new System.Drawing.Font("Arial", 72F,
System.Drawing.FontStyle.Regular), new Color(255, 0, 0));

document.Add(new Paragraph("Embedded Font: Arial, 72F, Rregular", aFont));

//  close document and save to the output file
document.Close();
```

# How to add an image

## Add an image to the document

```
String outputFilePath = Program.RootPath + "\\" + "Sample21.pdf";

String imageFilePath = Program.RootPath + "\\" + "Image001.bmp";

Document document = new Document();
PDFBuildHandler.Create(document, outputFilePath);

//  open document
document.Open();

document.Add(new Paragraph("Sample: add an image."));

//  load an image from the source file.
Image image = new Image(imageFilePath);
document.Add(image);

document.Add(new Paragraph("Image Width: " + image.Width + " pt.; Image Height: " + image.Height
+ " pt."));

//  close document and save to the output file
document.Close();
```

## Add an image with the specified size

```
String outputFilePath = Program.RootPath + "\\" + "Sample22.pdf";

String imageFilePath = Program.RootPath + "\\" + "Image001.bmp";

Document document = new Document();
PDFBuildHandler.Create(document, outputFilePath);

//  open document
document.Open();

document.Add(new Paragraph("Scale image size to: width = 500 pt.; height = 50 pt."));

Image image1 = new Image(imageFilePath);
//  scale image width to 500 pt. (72 ppi)
image1.ScaleAbsoluteWidth(500);
//  scale image height to 50 pt. (72 ppi)
image1.ScaleAbsoluteHeight(50);

document.Add(image1);

document.Add(new Paragraph("Scale image size to: width = 50 pt.; height = 200 pt."));

Image image2 = new Image(imageFilePath);
//  scale image width to 50 pt. (72 ppi)
image2.ScaleAbsoluteWidth(50);
//  scale image height to 200 pt. (72 ppi)
image2.ScaleAbsoluteHeight(200);

document.Add(image2);


//  close document and save to the output file
document.Close();
```

## Add a figure which created by the context

```
String outputFilePath = Program.RootPath + "\\" + "Sample24.pdf";

Document document = new Document();
PDFBuildHandler.Create(document, outputFilePath);

//  open document
document.Open();

document.Add(new Paragraph("Sample: add a figure"));

Figure figure = createFigureByContext();
document.Add(figure);

document.Add(new Paragraph("Figure Size: Width = " + figure.Width + " pt.; Height = " +
figure.Height + " pt."));

//  close document and save to the output file
document.Close();
private static Figure createFigureByContext()
{
    //  use PDFContext to design a figure
    PDFContext ctx = new PDFContext();
    //  set figure size: width 400 pixels; height 300 pixels
    ctx.SetWidth(new RELength(400, Units.PX));
    ctx.SetHeight(new RELength(300, Units.PX));
    //  set figure background color to Yellow
    ctx.SetBackGroundColor(System.Drawing.Color.Yellow);
    //  draw a red circle
    ctx.DrawEllipse(new REPen(new REColor(255, 0, 0)), 50, 50, 100, 100);

    //  create the figure by PDFContext
    return new Figure(ctx);
}
```

# How to apply list

## Add an unnumbered list

```
String outputFilePath = Program.RootPath + "\\" + "Sample31.pdf";

Document document = new Document();
PDFBuildHandler.Create(document, outputFilePath);

//  open document
document.Open();

//  initial a unnumbered list
List list = new List(false);
for (int i = 0; i < 5; i++)
{
    ListItem listItem = new ListItem("This is list item " + i + ".");
    list.Add(listItem);
}

//  add list to the document
document.Add(list);

//  close document and save to the output file
document.Close();
```

## Add a numbered list

```
String outputFilePath = Program.RootPath + "\\" + "Sample32.pdf";

Document document = new Document();
PDFBuildHandler.Create(document, outputFilePath);

//  open document
document.Open();

//  initial a numbered list
List list = new List(true);
for (int i = 0; i < 5; i++)
{
    ListItem listItem = new ListItem("This is list item " + i + ".");
    list.Add(listItem);
}
```

```
//  add list to the document
document.Add(list);

//  close document and save to the output file
document.Close();
```

## Use letter as list symbol for a list

```
String outputFilePath = Program.RootPath + "\\" + "Sample33.pdf";

Document document = new Document();
PDFBuildHandler.Create(document, outputFilePath);

//  open document
document.Open();

//  initial a unnumbered list
List list1 = new List(false, false);
for (int i = 0; i < 3; i++)
{
    ListItem listItem = new ListItem("Unnumbered list: Item " + i + ".");
    list1.Add(listItem);
}

//  initial a numbered list
List list2 = new List(true, false);
for (int i = 0; i < 3; i++)
{
    ListItem listItem = new ListItem("Numbered list: Item " + i + ".");
    list2.Add(listItem);
}

//  initial a unnumbered list
List list3 = new List(false, true);
for (int i = 0; i < 3; i++)
{
    ListItem listItem = new ListItem("Unnumbered list (Letter): Item " + i + ".");
    list3.Add(listItem);
}

//  initial a numbered list
List list4 = new List(true, true);
for (int i = 0; i < 3; i++)
{
    ListItem listItem = new ListItem("Numbered list (Letter): Item " + i + ".");
    list4.Add(listItem);
}


//  add lists to the document
document.Add(list1);
document.Add(list2);
document.Add(list3);
document.Add(list4);
```

```
//  close document and save to the output file
document.Close();
```

## Change symbol indent for a list

```
String outputFilePath = Program.RootPath + "\\" + "Sample34.pdf";

Document document = new Document();
PDFBuildHandler.Create(document, outputFilePath);

//  open document
document.Open();

float symbolIndent = 2F;
//  initial a numbered list
List list1 = new List(true, symbolIndent);
for (int i = 0; i < 3; i++)
{
    ListItem listItem = new ListItem("List item " + i + ".");
    list1.Add(listItem);
}

symbolIndent = 4F;
//  initial a numbered list
List list2 = new List(true, symbolIndent);
for (int i = 0; i < 3; i++)
{
    ListItem listItem = new ListItem("List item " + i + ".");
    list2.Add(listItem);
}

symbolIndent = 6F;
//  initial a numbered list
List list3 = new List(true, symbolIndent);
for (int i = 0; i < 3; i++)
{
    ListItem listItem = new ListItem("List item " + i + ".");
    list3.Add(listItem);
}

//  add list to the document
document.Add(list1);
document.Add(list2);
document.Add(list3);

//  close document and save to the output file
document.Close();
```

# How to add a table

## Create an empty table

```
String outputFilePath = Program.RootPath + "\\" + "Sample41.pdf";

Document document = new Document();
PDFBuildHandler.Create(document, outputFilePath);

//  open document
document.Open();

//  initial a table with 4 columns
PdfPTable table = new PdfPTable(4);
//  3 rows
for (int i = 0; i < 3; i++)
{   //  set cell height to 20 pt. for each row
    PdfPCell cell = new PdfPCell();
    cell.Height = 20F;
    table.AddCell(cell);
    //  finish remained columns of the current row
    table.CompleteRow();
}

//  add table to the document
document.Add(table);

//  close document and save to the output file
document.Close();
```

## Create a table with relative column width

```
String outputFilePath = Program.RootPath + "\\" + "Sample42.pdf";

Document document = new Document();
PDFBuildHandler.Create(document, outputFilePath);

//  open document
document.Open();

//  the width ratios of table columns -- 1 : 2 : 1 : 1
float[] columnRelativeWidths = new float[4] { 1, 2, 1, 1 };
PdfPTable table = new PdfPTable(columnRelativeWidths);
//  set lock column width flag to true
table.LockedWidth = true;
//  set total table width to 360 pt. (5 inches), by using above relative widths, the actural
column widths are:
//  72 pt., 144 pt., 72 pt., 72 pt. in turn.
table.TotalWidth = 360;

//  3 rows
for (int i = 0; i < 3; i++)
{   //  set cell height to 20 pt. for each row
    PdfPCell cell = new PdfPCell();
    cell.Height = 20F;
    table.AddCell(cell);
    //  finish remained columns of the current row
    table.CompleteRow();
}

//  add table to the document
document.Add(table);

//  close document and save to the output file
document.Close();
```

## Create a table with absolute column width

```
String outputFilePath = Program.RootPath + "\\" + "Sample43.pdf";

Document document = new Document();
PDFBuildHandler.Create(document, outputFilePath);

//  open document
document.Open();

//  4 columuns with widths; 72 pt., 144 pt., 72 pt., 72 pt.
float[] columnWidthInPoints = new float[4] { 72F, 144F, 72F, 72F };
PdfPTable table = new PdfPTable(columnWidthInPoints.Length);
//  set lock column width flag to true
table.LockedWidth = true;
table.SetTotalWidth(columnWidthInPoints);

//  3 rows
for (int i = 0; i < 3; i++)
{   //  set cell height to 20 pt. for each row
    PdfPCell cell = new PdfPCell();
    cell.Height = 20F;
    table.AddCell(cell);
    //  finish remained columns of the current row
    table.CompleteRow();
}

//  add table to the document
document.Add(table);

//  close document and save to the output file
document.Close();
```

## Set table alignment

```
String outputFilePath = Program.RootPath + "\\" + "Sample44.pdf";

Document document = new Document();
PDFBuildHandler.Create(document, outputFilePath);

//  open document
document.Open();

PdfPTable table1 = new PdfPTable(4);
//  set table to left alignment
table1.HorizontalAlignment = Alignment.ALIGN_LEFT;
for (int i = 0; i < 3; i++)
{   //  set cell height to 20 pt. for each row
    PdfPCell cell = new PdfPCell();
    cell.Height = 20F;
    table1.AddCell(cell);
    //  finish remained columns of the current row
    table1.CompleteRow();
}
document.Add(table1);

PdfPTable table2 = new PdfPTable(4);
//  set table to right alignment
table2.HorizontalAlignment = Alignment.ALIGN_RIGHT;
for (int i = 0; i < 3; i++)
{   //  set cell height to 20 pt. for each row
    PdfPCell cell = new PdfPCell();
    cell.Height = 20F;
    table2.AddCell(cell);
    //  finish remained columns of the current row
    table2.CompleteRow();
}
document.Add(table2);

PdfPTable table3 = new PdfPTable(4);
//  set table to center alignment
table3.HorizontalAlignment = Alignment.ALIGN_CENTER;
for (int i = 0; i < 3; i++)
{   //  set cell height to 20 pt. for each row
    PdfPCell cell = new PdfPCell();
    cell.Height = 20F;
    table3.AddCell(cell);
    //  finish remained columns of the current row
    table3.CompleteRow();
}
```

```
document.Add(table3);

//  close document and save to the output file
document.Close();
```

## Add table cell to a table

```
String outputFilePath = Program.RootPath + "\\" + "Sample45.pdf";

Document document = new Document();
PDFBuildHandler.Create(document, outputFilePath);

//  open document
document.Open();

//  initial a table with 4 columns
PdfPTable table = new PdfPTable(4);

//  add cell to the 1st row
table.AddCell("Row 1, Cell 1");     //  1st cell
table.AddCell("Row 1, Cell 2");     //  2nd cell
table.AddCell("Row 1, Cell 3");     //  3rd cell
table.AddCell("Row 1, Cell 4");     //  4th cell
//  finish remained columns of the current row, do nothing if there is no remained cell in the
row.
table.CompleteRow();

//  add cell to the 2nd row
table.AddCell("");                  //  1st cell is empty
table.AddCell("Row 2, Cell 2");     //  2nd cell
table.AddCell("");                  //  3rd cell is empty
//  finish remained columns of the current row
table.CompleteRow();

//  add cell to the 2nd row
table.AddCell("");                  //  1st cell is empty
table.AddCell("");                  //  2nd cell is empty
table.AddCell("Row 3, Cell 3");     //  3rd cell
//  finish remained columns of the current row
table.CompleteRow();

//  add table to the document
document.Add(table);

//  close document and save to the output file
document.Close();
```

## Create a simple calendar

```
String outputFilePath = Program.RootPath + "\\" + "Sample46.pdf";

Document document = new Document();
PDFBuildHandler.Create(document, outputFilePath);

//  open document
document.Open();

float[] columnWidths = new float[7] { 72, 72, 72, 72, 72, 72, 72 };
PdfPTable table = new PdfPTable(columnWidths.Length);
table.LockedWidth = true;
table.SetTotalWidth(columnWidths);

Font titleFont = new Font(Font.FontFamily.Helvetica, 12F, Font.FontStyle.Bold, new Color(0, 0,
255));
String[] colNames = new String[7] { "Sunday", "Monday", "Tuesday", "Wednesday", "Thursday",
"Friday", "Saturday" };
foreach (String name in colNames)
    table.AddCell(createCell(name, titleFont, 16F));
//table.CompleteRow();

Font weekdayFont = new Font(Font.FontFamily.Helvetica, 36F, Font.FontStyle.Regular, new Color(0,
0, 0));
Font weekendFont = new Font(Font.FontFamily.Helvetica, 36F, Font.FontStyle.Bold, new Color(255,
0, 0));
int date = 1;
while (date < 31)
{
    for (int col = 0; col < 7; col++)
    {
        Font dateFont = (col == 0 || col == 6) ? weekendFont : weekdayFont;
        table.AddCell(createCell(date.ToString(), dateFont, 40F));

        if (++date > 31) break;
    }
    table.CompleteRow();
```

```
}

//  add content to the document
document.Add(table);

//  close document and save to the output file
document.Close();
private static PdfPCell createCell(String content, Font font, float cellHeight)
{
    Paragraph p = new Paragraph(content, font);
    p.Alignment = Alignment.ALIGN_CENTER;
    PdfPCell cell = new PdfPCell(p);
    cell.Height = cellHeight;
    return cell;
}
```

# How to use chapter and section

## Create a document with three empty chapters

```
String outputFilePath = Program.RootPath + "\\" + "Sample61.pdf";

Document document = new Document();
PDFBuildHandler.Create(document, outputFilePath);

document.Open();

//  add a chapter that title is "Chapter 1"
document.Add(new Chapter("Chapter 1", 1));
//  add a chapter that title is "Chapter 2"
document.Add(new Chapter("Chapter 2", 2));
//  add a chapter that title is "Chapter 3"
document.Add(new Chapter("Chapter 3", 3));

document.Close();
```

## Add sections to a chapter

```
String outputFilePath = Program.RootPath + "\\" + "Sample62.pdf";

Document document = new Document();
PDFBuildHandler.Create(document, outputFilePath);

document.Open();

Chapter chapter = new Chapter("Chapter: Sample", 2);

//  add section
chapter.AddSection("Section 1");
//  add section
chapter.AddSection("Section 2");
//  add section
chapter.AddSection("Section 3");


document.Add(chapter);

document.Close();
```

## Change title string shown in the bookmark entry

```
String outputFilePath = Program.RootPath + "\\" + "Sample63.pdf";

Document document = new Document();
PDFBuildHandler.Create(document, outputFilePath);

document.Open();

Chapter chapter = new Chapter("This is a sample chapter", 2);
//  change title string in bookmark entry
chapter.SetBookMarkTitle("Chapter");

//  add section
Section section1 = chapter.AddSection("Introduction");
//  change title string in bookmark entry
section1.SetBookMarkTitle("bookmark entry 1");
//  change title string in bookmark entry
Section section2 = chapter.AddSection("Main");
section2.SetBookMarkTitle("bookmark entry 2");
//  change title string in bookmark entry
Section section3 = chapter.AddSection("Discussion");
section3.SetBookMarkTitle("bookmark entry 3");
//  change title string in bookmark entry
Section section4 = chapter.AddSection("Conclusion");
section4.SetBookMarkTitle("bookmark entry 4");

document.Add(chapter);

document.Close();
```

## Add nested sections to chapter

```
String outputFilePath = Program.RootPath + "\\" + "Sample64.pdf";

Document document = new Document();
PDFBuildHandler.Create(document, outputFilePath);

document.Open();

Font chapter_font = new Font(Font.FontFamily.TimesRoman, 24F, Font.FontStyle.Bold, Color.Blue);
Font section_level_1_font = new Font(Font.FontFamily.TimesRoman, 16F, Font.FontStyle.Regular);
Font section_level_2_font = new Font(Font.FontFamily.TimesRoman, 14F, Font.FontStyle.Bold);
Font section_level_3_font = new Font(Font.FontFamily.TimesRoman, 14F, Font.FontStyle.Italic);
Font body_font = new Font(Font.FontFamily.TimesRoman, 12F, Font.FontStyle.Regular);

Chapter chapter = new Chapter(new Paragraph("This is a sample chapter", chapter_font), 2);

//  Section 1
Section section1 = chapter.AddSection(new Paragraph("Introduction", section_level_1_font));
section1.Add(new Paragraph("Paragraph in section 1", body_font));

//  Section 1.1
Section section1_1 = section1.AddSection(new Paragraph("Section A", section_level_2_font));
section1_1.Add(new Paragraph("Paragraph in section 1.1", body_font));

//  Section 1.2
Section section1_2 = section1.AddSection(new Paragraph("Section B", section_level_2_font));
section1_2.Add(new Paragraph("Paragraph in section 1.2", body_font));

//  Section 2
Section section2 = chapter.AddSection(new Paragraph("Main", section_level_1_font));
section2.Add(new Paragraph("Paragraph in section 2", body_font));

//  Section 3
Section section3 = chapter.AddSection(new Paragraph("Discussion", section_level_1_font));

//  Section 3.1
Section section3_1 = section3.AddSection(new Paragraph("Section A", section_level_2_font));
section3_1.Add(new Paragraph("Paragraph in section 3.1", body_font));

//  Section 3.1.1
Section section3_1_1 = section3_1.AddSection(new Paragraph("Section i", section_level_3_font));
section3_1_1.Add(new Paragraph("Paragraph in section 3.1.1", body_font));

//  Section 4
Section section4 = chapter.AddSection("Conclusion");
```

```
document.Add(chapter);

document.Close();
```

# How to add Header and Footer

## Add a simple header/footer

```
C#
String outputFilePath = Program.RootPath + "\\" + "Sample71.pdf";

Document document = new Document();
PDFBuildHandler.Create(document, outputFilePath);

document.Open();

//  create a header
Header header = new Header();
header.SetOptions(HeaderFooterOptions.All);
//  set header content
Paragraph headerContent = new Paragraph("This is a header");
//  set header alignment
headerContent.Alignment = Alignment.ALIGN_CENTER;
header.AddParagraph(0, headerContent);

//  create a footer
Footer footer = new Footer();
footer.SetOptions(HeaderFooterOptions.All);
//  set footer content
Paragraph footerContent = new Paragraph("This is a footer");
//  set footer alignment
footerContent.Alignment = Alignment.ALIGN_RIGHT;
footer.AddParagraph(0, footerContent);

//  set header to the document
document.SetHeader(header);
//  set footer to the document
document.SetFooter(footer);

document.Add(new Paragraph("This is body field."));

//  add a new page (the 2nd page)
document.NewPage();
//  add a new page (the 3rd page)
document.NewPage();

document.Close();
```

# How to add Footnote and Endnote

## Add a simple footnote

```
C#
String outputFilePath = Program.RootPath + "\\" + "Sample81.pdf";

Document document = new Document();

PDFBuildHandler.Create(document, outputFilePath);

document.Open();
document.SetFootnoteInfo(new FootnoteSetting());

Paragraph p = new Paragraph();
p.AddChunk(new Chunk("This is a sample for footnote"));

//  insert a footnote
Footnote footnote1 = new Footnote("This is the 1st footnote");
p.AddChunk(footnote1);

p.AddChunk(new Chunk("."));

p.AddChunk(new Chunk("This is a sample for the 2nd footnote"));

//  insert a footnote
Footnote footnote2 = new Footnote("This is the 2nd footnote");
p.AddChunk(footnote2);

p.AddChunk(new Chunk("."));

document.Add(p);

document.Close();
```

## Add a simple endnote

```
C#
String outputFilePath = Program.RootPath + "\\" + "Sample82.pdf";

Document document = new Document();

PDFBuildHandler.Create(document, outputFilePath);

document.Open();
document.SetEndnoteInfo(new EndnoteSetting());

Paragraph p = new Paragraph();

p.AddChunk(new Chunk("This is a sample for endnote"));

//  insert a endnote
Endnote endnote1 = new Endnote("This is the 1st endnote");
p.AddChunk(endnote1);

p.AddChunk(new Chunk(". This is a sample for the 2nd endnote"));

//  insert a endnote
Endnote endnote2 = new Endnote("This is the 2nd endnote");
p.AddChunk(endnote2);

p.AddChunk(new Chunk("."));

document.Add(p);

//  insert a new page
document.NewPage();

document.Add(new Paragraph("This is the 2nd page."));

document.Close();
```

# How to add form fields

## Add  text box form field

C#

```
String outputFilePath = Program.RootPath + "\\" + "Sample91.pdf";

Document document = new Document();

PDFBuildHandler.Create(document, outputFilePath);

document.Open();

document.Add(new Paragraph("This is a sample for Text Form Field:"));
document.Add(new Paragraph(" "));

//  create a Text Box form field
FormFieldTextBox textBox = new FormFieldTextBox("TextBox1");
textBox.Text = " ";
textBox.Width = 400;
textBox.Height = 100;
//  add form field
document.Add(textBox);

document.Close();
```

## Add check box form field

C#

```
String outputFilePath = Program.RootPath + "\\" + "Sample92.pdf";

Document document = new Document();

PDFBuildHandler.Create(document, outputFilePath);

document.Open();

document.Add(new Paragraph("This is a sample for Text Form Field:"));
document.Add(new Paragraph(" "));

//  create a Text Box form field
FormFieldCheckBox checkbox = new FormFieldCheckBox("CheckBox1");
checkbox.Text = " ";
checkbox.Width = 50;
checkbox.Height = 50;
//  add form field
document.Add(checkbox);

document.Close();
```

# How to use table template to create a table

## Create a simple table by using table template

Step 1: design data class for customer data

```
public class Sample1TableDef
{
    public const String WeeklySalesRecord_ProductNo = "Product No";
    public const String WeeklySalesRecord_ProductCode = "Product Code";
    public const String WeeklySalesRecord_ProductName = "Product Name";
    public const String WeeklySalesRecord_MondayCount = "Monday Count";
    public const String WeeklySalesRecord_TuesdayCount = "Tuesday Count";
    public const String WeeklySalesRecord_WednesdayCount = "Wednesday Count";
    public const String WeeklySalesRecord_ThursdayCount = "Thursday Count";
    public const String WeeklySalesRecord_FridayCount = "Friday Count";
    public const String WeeklySalesRecord_SaturdayCount = "Saturday Count";
    public const String WeeklySalesRecord_SundayCount = "Sunday Count";
    public const String WeeklySalesRecord_TotalCount = "Total Count";


    public const String StyleID_Body_Default = "BodyDefault";
    public const String StyleID_Head_Default = "HeaderDefault";

    public static String GetHeaderType(PropEntry entry)
    {
        return Sample1TableDef.StyleID_Head_Default;
    }
}
public class WeeklySalesRecord : ValueEntry
{
    private String _num;
    private String _code;
    private String _product;

    private String _mon;
    private String _tue;
    private String _wed;
    private String _thu;
    private String _fri;
    private String _sat;
    private String _sun;

    private String _total;
```

```
    public WeeklySalesRecord(String num, String code, String product,
                             String mon, String tue, String wed, String thu, String fri, String
sat, String sun,
                             String total)
        : base()
    {

        this._num = num;
        this._code = code;
        this._product = product;
        this._mon = mon;
        this._tue = thu;
        this._wed = wed;
        this._thu = thu;
        this._fri = fri;
        this._sat = sat;
        this._sun = sun;
        this._total = total;

        AddPropertyEntry(Sample1TableDef.WeeklySalesRecord_ProductNo, this._num, 1);
        AddPropertyEntry(Sample1TableDef.WeeklySalesRecord_ProductCode, this._code, 1);
        AddPropertyEntry(Sample1TableDef.WeeklySalesRecord_ProductName, this._product, 1);
        AddPropertyEntry(Sample1TableDef.WeeklySalesRecord_MondayCount, this._mon, 1);
        AddPropertyEntry(Sample1TableDef.WeeklySalesRecord_TuesdayCount, this._tue, 1);
        AddPropertyEntry(Sample1TableDef.WeeklySalesRecord_WednesdayCount, this._wed, 1);
        AddPropertyEntry(Sample1TableDef.WeeklySalesRecord_ThursdayCount, this._thu, 1);
        AddPropertyEntry(Sample1TableDef.WeeklySalesRecord_FridayCount, this._fri, 1);
        AddPropertyEntry(Sample1TableDef.WeeklySalesRecord_SaturdayCount, this._sat, 1);
        AddPropertyEntry(Sample1TableDef.WeeklySalesRecord_SundayCount, this._sun, 1);
        AddPropertyEntry(Sample1TableDef.WeeklySalesRecord_TotalCount, this._total, 1);
    }

    public override string GetStyleID(PropEntry entry, TableState state)
    {
        return Sample1TableDef.StyleID_Body_Default;
    }
}
```

Step 2: prepare table template configure file (.xml)

```
<?xml version="1.0" encoding="utf-8" ?>
<TableTemplate>
  <Settings TableWidth="500" RelativeWidth="0.04 0.08 0.15 0.09 0.09 0.09 0.09 0.09 0.09 0.09
0.10" />
  <Header>
    <Property Name="Product No" Content="No" ColspanNum="1" RowspanNum="1" />
    <Property Name="Product Code" Content="Code" ColspanNum="1" RowspanNum="1" />
    <Property Name="Product Name" Content="Product" ColspanNum="1" RowspanNum="1" />
    <Property Name="Monday Count" Content="Mon" ColspanNum="1" RowspanNum="1" />
    <Property Name="Tuesday Count" Content="Tue" ColspanNum="1" RowspanNum="1" />
    <Property Name="Wednesday Count" Content="Wed" ColspanNum="1" RowspanNum="1" />
    <Property Name="Thursday Count" Content="Thu" ColspanNum="1" RowspanNum="1" />
    <Property Name="Friday Count" Content="Fri" ColspanNum="1" RowspanNum="1" />
    <Property Name="Saturday Count" Content="Sat" ColspanNum="1" RowspanNum="1" />
    <Property Name="Sunday Count" Content="Sun" ColspanNum="1" RowspanNum="1" />
    <Property Name="Total Count" Content="Total" ColspanNum="1" RowspanNum="1" />
  </Header>
  <Styles>
    <Style ID="BodyDefault">
      <FontFamily>Helvetica</FontFamily>
      <FontSize>10</FontSize>
      <!--FontStyle valid value: [Regular|Bold|Italic|BoldItalic]-->
      <FontStyle>Regular</FontStyle>
      <FontColor>#000000</FontColor>
      <BorderLineWidth>0.5</BorderLineWidth>
      <BackGroundColor>#FFFFFF</BackGroundColor>
      <HoriAlignment>Center</HoriAlignment>
      <VertAlignment>Middle</VertAlignment>
      <IsTopLineExit>True</IsTopLineExit>
      <IsBottomLineExit>True</IsBottomLineExit>
      <IsLeftLineExit>True</IsLeftLineExit>
      <IsRightLineExit>True</IsRightLineExit>
      <LineType>Solid</LineType>
    </Style>
    <Style ID="HeaderDefault">
      <FontFamily>Helvetica</FontFamily>
      <FontSize>12</FontSize>
      <!--FontStyle valid value: [Regular|Bold|Italic|BoldItalic]-->
      <FontStyle>Bold</FontStyle>
      <FontColor>#000000</FontColor>
      <BorderLineWidth>0.5</BorderLineWidth>
      <BackGroundColor>#C1CCE8</BackGroundColor>
      <HoriAlignment>Center</HoriAlignment>
      <VertAlignment>Middle</VertAlignment>
      <IsTopLineExit>True</IsTopLineExit>
      <IsBottomLineExit>True</IsBottomLineExit>
      <IsLeftLineExit>True</IsLeftLineExit>
      <IsRightLineExit>True</IsRightLineExit>
      <LineType>Solid</LineType>
    </Style>
  </Styles>
</TableTemplate>
```

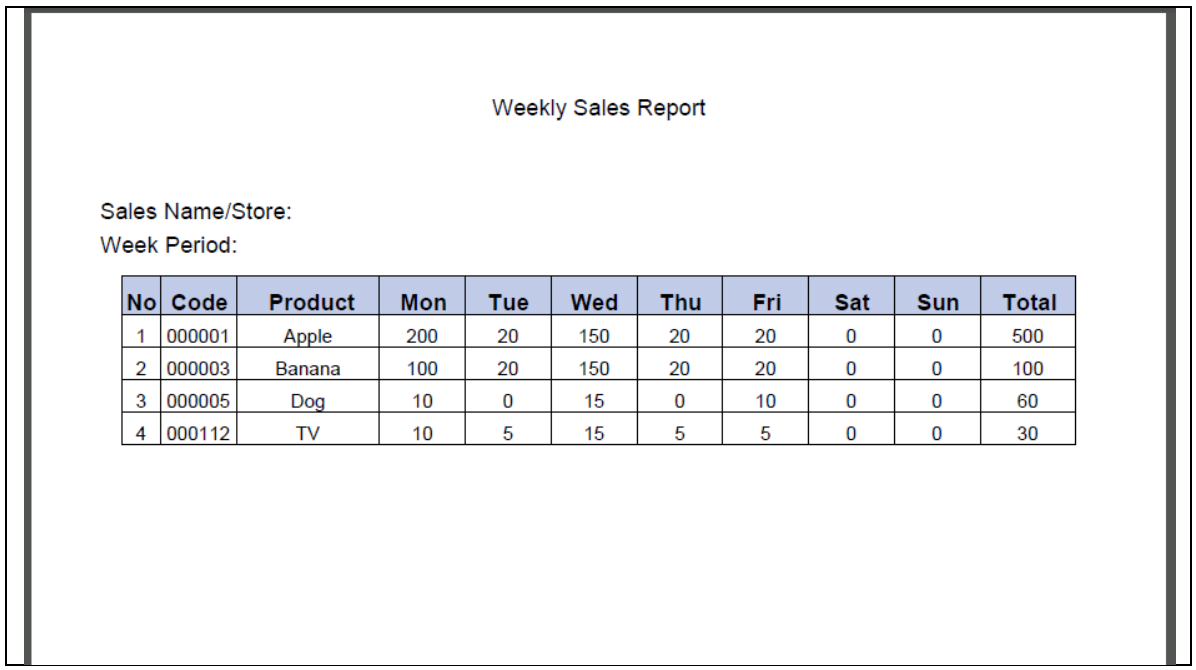
Step 3: prepare sample data

```
public class Sample1Database
{

public static String[] datas = new String[] {
    "1;000001;Apple;200;200;150;20;20;0;0;500",
    "2;000003;Banana;100;50;150;20;20;0;0;100",
    "3;000005;Dog;10;20;15;0;10;0;0;60",
    "4;000112;TV;10;20;15;5;5;0;0;30",
};


public static List<ValueEntry> CreateTableEntries()
{
    List<ValueEntry> values = new List<ValueEntry>();

    foreach (String data in datas)
    {
        String[] tmps = data.Split(new char[] { ';' }, StringSplitOptions.RemoveEmptyEntries);
        if (tmps.Length < 11) continue;

        values.Add(new WeeklySalesRecord(tmps[0], tmps[1], tmps[2],
                                         tmps[3], tmps[4], tmps[5], tmps[6], tmps[7], tmps[8],
tmps[9],
                                         tmps[10]));
    }

    return values;
}
}
```

Step 4: create document with the table

```
String outputFilePath = Program.OutputFolder + "\\" + OutputFileName;

Document doc = new Document();
PDFBuildHandler.Create(doc, outputFilePath);

doc.Open();

Paragraph p = new Paragraph("Weekly Sales Report");
p.Alignment = Alignment.ALIGN_CENTER;
p.TextFont = new Font(Font.FontFamily.TimesRoman, 36F, Font.FontStyle.Bold, Color.Black);
p.SpacingAfter = 36;
doc.Add(p);

Paragraph p1 = new Paragraph("Sales Name/Store: ");
p1.TextFont = new Font(Font.FontFamily.Helvetica, 16F, Font.FontStyle.Regular, Color.Black);
doc.Add(p1);

Paragraph p2 = new Paragraph("Week Period: ");
p2.TextFont = new Font(Font.FontFamily.Helvetica, 16F, Font.FontStyle.Regular, Color.Black);
p2.SpacingAfter = 12;
doc.Add(p2);

addTable(doc);

doc.Close();
private static void addTable(Document doc)
{
    String xmlFilePath = Program.OutputFolder + "\\" + TableTemplateXMLFileName;
    TabelTemplate tableTemplate = TabelTemplate.LoadTemplate(xmlFilePath);
    tableTemplate.SetHeaderPropertyFunc(Sample1TableDef.GetHeaderType);

    doc.SetTableTemplate(tableTemplate);

    List<ValueEntry> values = Sample1Database.CreateTableEntries();
    foreach (ValueEntry value in values)
    {
        doc.AddTableEntry(value);
    }

    doc.CloseCurrentTable();
}
```

Output:



Weekly Sales Report

Sales Name/Store:

Week Period:

| No | Code | Product | Mon | Tue | Wed | Thu | Fri | Sat | Sun | Total |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 000001 | Apple | 200 | 20 | 150 | 20 | 20 | 0 | 0 | 500 |
| 2 | 000003 | Banana | 100 | 20 | 150 | 20 | 20 | 0 | 0 | 100 |
| 3 | 000005 | Dog | 10 | 0 | 15 | 0 | 10 | 0 | 0 | 60 |
| 4 | 000112 | TV | 10 | 5 | 15 | 5 | 5 | 0 | 0 | 30 |