Fixed and Flow document layout in Apitron PDF Kit .NET component

Written by Apitron Documentation Team

Introduction

Apitron PDF Kit is a .NET component that makes you able to create, edit, combine, split, sign and do whatever you want with PDF documents. It offers full support for PDF forms and advanced PDF graphics along with the easy to use and clean API. It's a cross platform pdf library that can be used to create applications for all modern mobile, desktop, web or cloud platforms. Read more about it here or download free book available by the following link.

This article is intended to give you brief overview of the API and explain the difference between two approaches which the library offers for PDF generation and manipulation. The topic was deeply covered in our book "Apitron PDF Kit in Action" linked above, and this article is a quick and packed summary of the concepts from this book.

PDF and Fixed Layout

"Classical" approach for PDF generation and creation, everyone is used to, is based on structures and entities defined by PDF specification. So most libraries offer APIs which are based on these low-level definitions, with an attempt to provide more high level abstraction depending on own architectural decision made by the developer.

E.g. a sequence of drawing operations can be called a `GraphicsPath` to make it look like the one from Windows GDI, and one would be required to instantiate this path object and add commands to it. Others call such objects Shapes or Paths. Same approach is used to name various routines, e.g. you could find `DrawString`, `DrawText` or `SaveState` in almost all such APIs.

While it's all different it's stays the same – it makes you able to create PDF documents based on absolute coordinates for all objects and that's why we call it a *fixed layout*. We offer this classic model as well, with a few exceptions described below:

- We tried to avoid the introduction of objects which are not part of PDF specification, therefore we didn't map the API to any known graphics system except defined by PDF. And we did it only if it was absolutely necessary. As a result, PDF specification can be used to look up things and our library to generate them. API follows it very strictly, overcoming all possible limitations, which might have been introduced by producing a PDF version of some well-known drawing API. PDF is an extremely complex format with rich graphical component and it's hard to find a universal way to work with it.

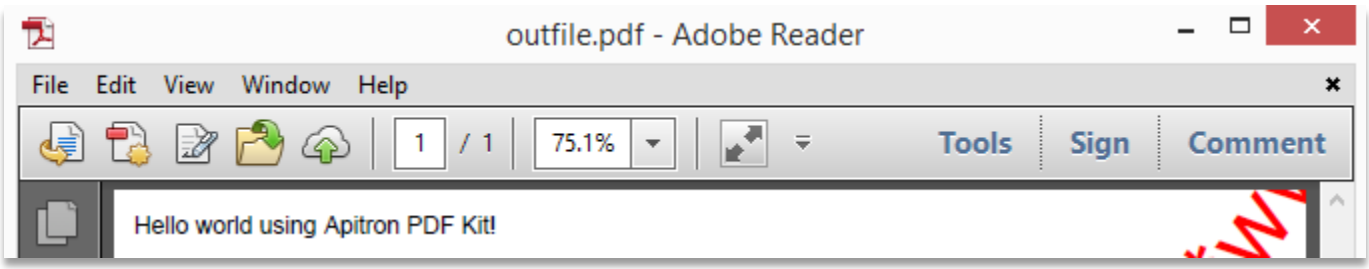
- We isolated the binary specifics of the PDF and, as a consequence, developers are not required to manage the binary part of format. The specification remains a good source of information about PDF features, objects appearance and their logical structure without the need to know how-tos related to binary form. You create PDF objects, move them around, copy etc. and library manages these, sometimes tricky, tasks and keeps everything consistent and readable.

This subset of Apitron PDF Kit API is located in `Apitron.PDF.Kit.FixedLayout` namespace and PDF document object is represented by `FixedDocument` class. Using this class and related fixed layout objects one may do whatever he or she needs with PDF document. PDF specification is fully covered by this subset.

Let’s check out how to create “Hello world” sample using fixed layout API:

```
// create output PDF file
using (FileStream outputStream = new FileStream("outfile.pdf", FileMode.Create, FileAccess.Write))
{
   // create new document
   FixedDocument document = new FixedDocument();

   // add blank first page
   document.Pages.Add(new Page(Boundaries.A4));

   // create text object and append text to it
   TextObject textObject = new TextObject(StandardFonts.Helvetica,12);

   // apply identity matrix, that doesn't change default appearance
   textObject.SetTextMatrix(1,0,0,1,0,0);
   textObject.AppendText("Hello world using Apitron PDF Kit!");

   // set current transformation matrix so text will be added to the top of the page,
   // PDF coordinate system has Y-axis directed from bottom to top.
   document.Pages[0].Content.Translate(10, 820);

   // add text object to page content, it will automatically create text showing operators
   document.Pages[0].Content.AppendText(textObject);

   // save to output stream
   document.Save(outputStream);
}
```

The result will look like this:



Pic. 1 FixedDocument, “Hello world” sample.

PDF and Flow Layout

In addition to “classic” approach described in section PDF and Fixed Layout, Apitron PDF Kit offers API subset called Flow Layout. It’s different from the fixed alternative being completely styles driven and using non-absolute element positioning approach. The closest alternative which can be seen is HTML + CSS.

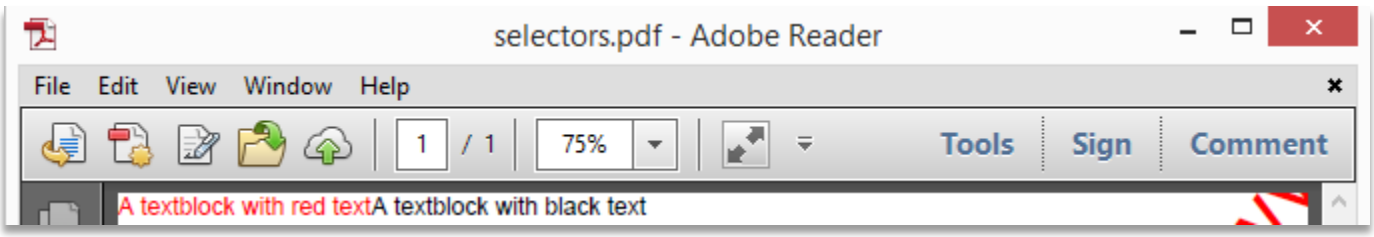
In flow layout API subset PDF document is represented by the class `FlowDocument` and all flow layout-related classes can be found in `Apitron.PDF.Kit.FlowLayout` namespace. Styles and corresponding objects are defined in `Apitron.PDF.Kit.Styles` namespace.

The general approach for creating a flow layout document is to construct it from various pre-defined content elements, e.g. text blocks, sections, grids, form controls, line breaks. Each of these elements can be styled either individually or by a matching style. Styles are assigned using specific selectors, so one may assign, for example, the border color to all text blocks in document using a single style with type selector.

See the styling code sample provided below:

```
using (Stream outputStream = File.Create("selectors.pdf"))
{
   // create new flow document
   FlowDocument document = new FlowDocument();
   // register style that matches all textblocks and sets their text color
   document.StyleManager.RegisterStyle("textblock", new Style(){Color = RgbColors.Red});
   // add textual content element
   document.Add(new TextBlock("A textblock with red text"));
   // add textual content element and directly set its property overriding the matched style
   document.Add(new TextBlock("A textblock with black text"){Color = RgbColors.Black});
   // save to output stream with page size A4
   document.Write(outputStream, new ResourceManager(), new PageBoundary(Boundaries.A4));
}
```

Resulting file looks as follows:



Pic. 2 FlowDocument, TextBlock styling sample

Supported features

Using Apitron PDF Kit you can do the following (the component is being regularly updated so the list may be incomplete):

- Create new PDF file from scratch or by copying content from other files
- Merge two or more documents into single PDF file
- Split PDF documents to separate pages
- Add, modify or extract content from PDF documents including text, images or attachments
- Work with transparency and complex graphics in PDF documents
- Create new or fill existing PDF forms , create or remove custom fields
- Check resources stored inside the PDF file, e.g. fonts, embedded files
- Digitally sign or check existing signatures present in PDF documents
- Protect document with a password or set permissions
- Work with navigation objects, e.g. create bookmarks or links
- Annotate document using all supported annotation objects
- Add various actions e.g. JavaScript or GoTo actions
- Search text in PDF files
- Use any font which is declared as supported in PDF specification
- Use and embed custom fonts
- Create custom Type3 fonts
- Use colors defined in all available colorspaces, e.g. Lab, XYZ, CMYK
- Control PDF viewer behavior using custom preview settings
- Save resulting files using PDF sub-standards e.g. PDF/A.
- Create cross-platform code which works on Windows, Windows Phone, Android and iOS (using Xamarin), Mac OS (using Mono) and any system Mono exists for.
- Create cloud PDF processing applications

The component is available for download by the following link. We'll be happy to answer your questions and welcome any feedback.