Adding image and text annotations to PDF documents, an interactive WPF PDF viewer sample

Written by Apitron Documentation Team

Introduction

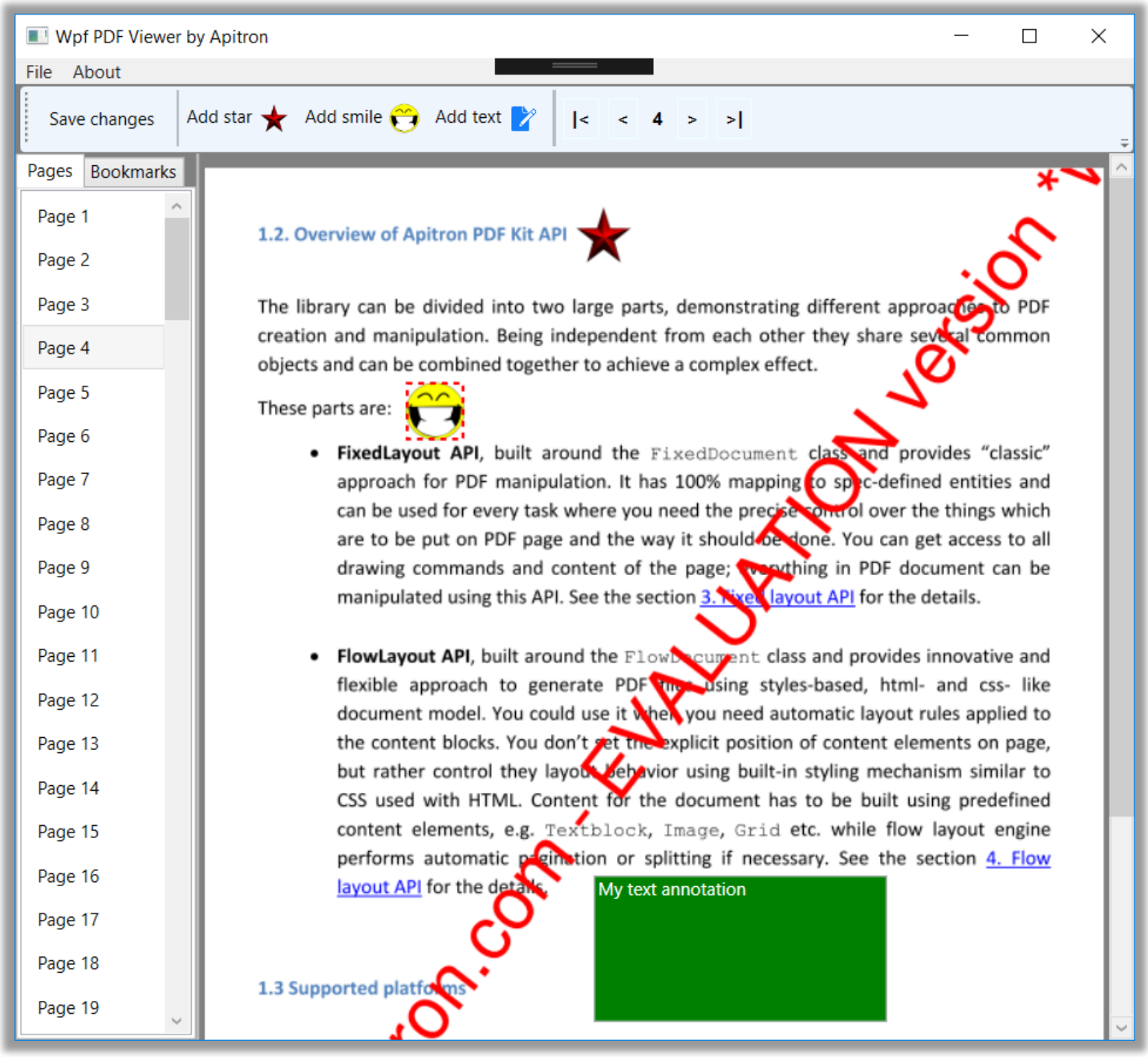
PDF annotations became the natural way to amend existing PDF documents, be it reviewer's notes, stamps or just custom images like approval marks. Natively supported they provide an easy and intuitive solution for collaboration.

Some PDF viewing programs provide support for annotations, but when it comes to custom software solutions it's sometimes needed to create specifically designed viewer control with annotations support.

In this article we'll show some highlights from the Wpf Pdf Viewer sample, giving you an idea of how it could look like to implement such a thing on your own. The sample uses two products: Apitron PDF Rasterizer for .NET for rendering and Apitron PDF Kit for .NET for PDF manipulation. It's not a complete solution, however it may be a good base for starting yours. The complete code sample can be downloaded from our github repo, it's called WpfPdfViewer.

The code

Our application's layout resembles a typical viewing application with navigation tab on the left and page area on the right. There's a toolbar with commands for adding various annotations and saving the final result. See the image below:



Pic. 1 Wpf PDF Viewer based on Apitron components

As it can be seen, it supports bookmarks and adding of several annotation types. Technically "smile" and "star" annotations are implemented using *RubberStampAnnotation* and use images, while the green one is a *FreeTextAnnotation*. These terms are from PDF specification and you'll find all the technical details there if needed. Our API follows this specification closely, so it won't be a problem to find a match between them. You can drag items and also edit the text. Selected one will be highlighted with the red animated border.

Core class here is a *DocumentViewModel*, it serves as a binding source in many places across the main window. Since we use MVVM, all other classes like *PageViewModel* and *AnnotationViewModel* are used to provide access to relevant page and annotation properties as well.

Navigation is implemented using Document's class Navigator object; it's related to Apitron PDF Rasterizer API and is provided to facilitate navigation implementation in custom viewers.

Custom view models are created for star, smile and text annotations. They can be found in *Annotations* subfolder of the solution.

When you click "Save changes" the document becomes saved and reloaded showing newly added annotations.

Sample code for creating the "star" annotation:

```
Annotation starAnnotation = new RubberStampAnnotation(new Boundary(10, 400, 50, 440),
    AnnotationFlags.Default);

FixedContent fixedContent = new FixedContent("ap01", new Boundary(0, 0, 40, 40));
fixedContent.Content.AppendImage("star", 0, 0, 40, 40);
starAnnotation.Appearance.Normal = fixedContent;
this.document.NativePage.Annotations.Add(starAnnotation);
```

Here we create an annotation object, set its appearance and add to the FixedDocument instance. The name "star" used for adding image is the name of the registered PDF resource, see the constructor of the MainWindow where such objects are registered.

The complete code sample can be downloaded from our github repo.

Summary

Custom PDF viewer capable of handling things like annotations can be easily created using our PDF components - Apitron PDF Kit and Apitron PDF Rasterizer. Surely, it becomes a PDF editor at some point, and as our components are available for many platforms like iOS and Andoid (via Xamarin), .NET, MONO, Windows Runtime, Windows Phone you can create cross platform apps with built-in PDF editors targeting all of them using the same code and same API.