How to create PDF document using MONO and Apitron PDF Kit

Written by Apitron Documentation Team

Introduction

To demonstrate the cross-platform nature of Apitron PDF Kit library we decided to create a small sample showing how one can programmatically create a PDF document using MONO and our pdf .NET component. You'll be able to run it on any MONO powered system, and we did it on OS X and Windows.

Prerequisites

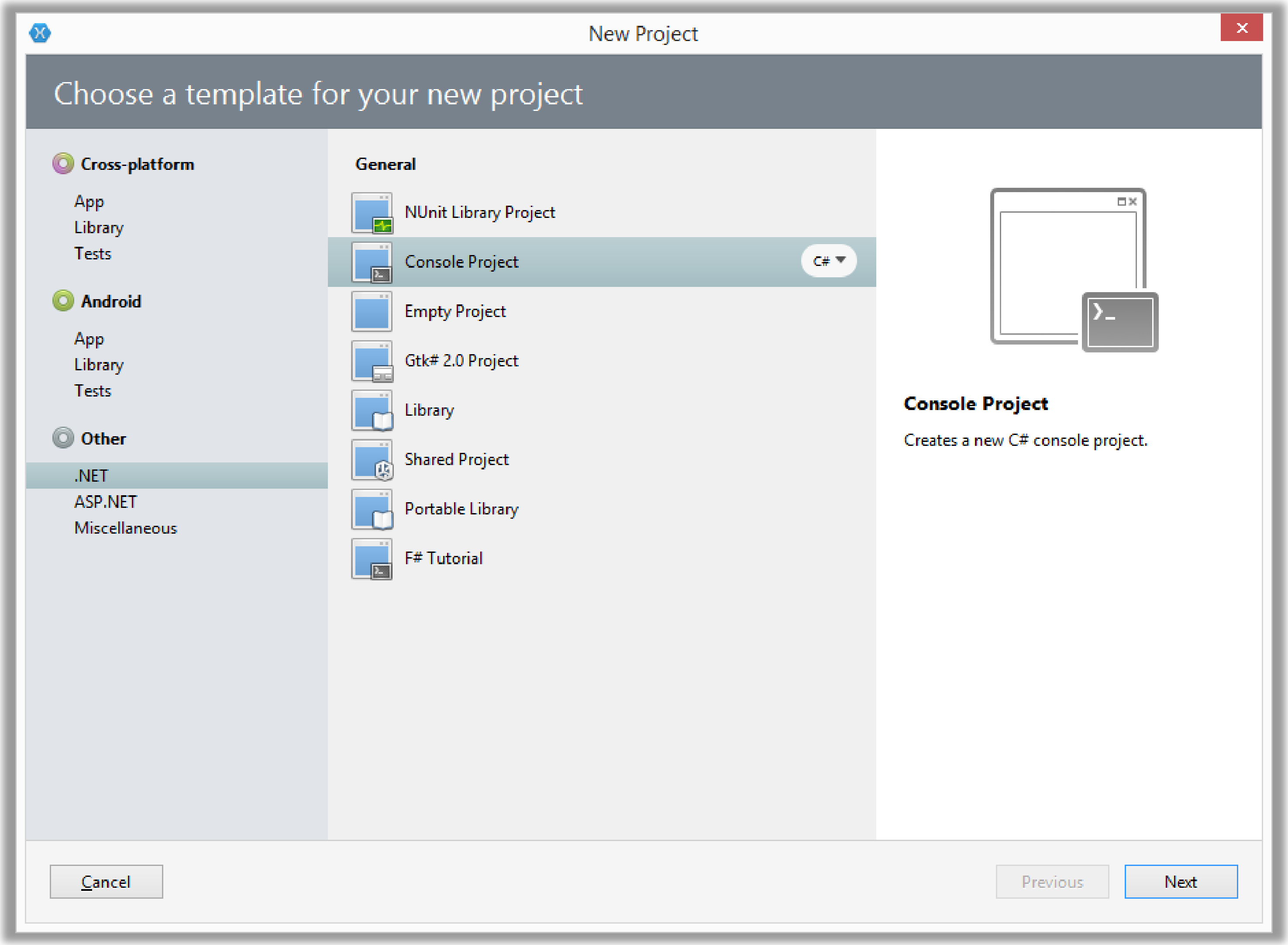
1) Make sure MONO is installed on all machines where you're going to run the app. We used v.4.0.3 available from mono project website.

2) This sample was created using Xamarin Studio (former MonoDevelop), so please download and install it as well.

3) Get the latest version of the Apitron PDF Kit for .NET component, you can download it here.

Creating the project

Open Xamarin Studio and select File -> New...- >Solution... -> .NET -> Console Project, as shown on the image below:



Console Project

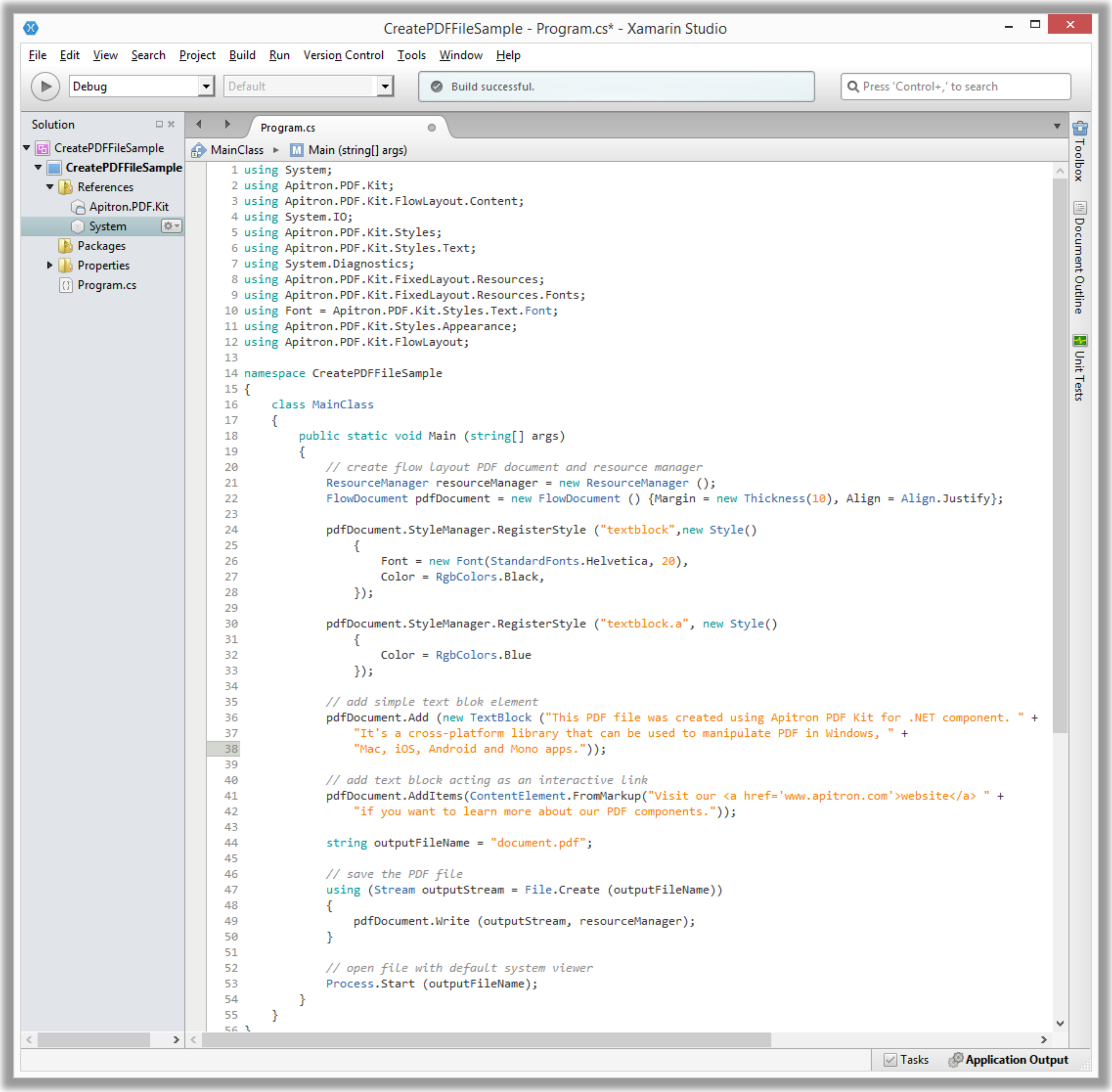
Creates a new C# console project.

Pic. 1 Creating new Mono.NET console project

After clicking “Next”, configure you project’s location and finish the creation.

Adding references

Add the reference to the **Apitron.PDF.Kit.dll** from the downloaded zip package, so your project references would look like on the image below:



```
using System;
using Apitron.PDF.Kit;
using Apitron.PDF.Kit.FlowLayout.Content;
using System.IO;
using Apitron.PDF.Kit.Styles;
using Apitron.PDF.Kit.Styles.Text;
using System.Diagnostics;
using Apitron.PDF.Kit.FixedLayout.Resources;
using Apitron.PDF.Kit.FixedLayout.Resources.Fonts;
using Font = Apitron.PDF.Kit.Styles.Text.Font;
using Apitron.PDF.Kit.Styles.Appearance;
using Apitron.PDF.Kit.FlowLayout;

namespace CreatePDFFileSample
{
    class MainClass
    {
        public static void Main (string[] args)
        {
            // create flow layout PDF document and resource manager
            ResourceManager resourceManager = new ResourceManager ();
            FlowDocument pdfDocument = new FlowDocument () {Margin = new Thickness(10), Align = Align.Justify};

            pdfDocument.StyleManager.RegisterStyle ("textblock",new Style()
                {
                    Font = new Font(StandardFonts.Helvetica, 20),
                    Color = RgbColors.Black,
                });

            pdfDocument.StyleManager.RegisterStyle ("textblock.a", new Style()
                {
                    Color = RgbColors.Blue
                });

            // add simple text blok element
            pdfDocument.Add (new TextBlock ("This PDF file was created using Apitron PDF Kit for .NET component. " +
                "It's a cross-platform library that can be used to manipulate PDF in Windows, " +
                "Mac, iOS, Android and Mono apps."));

            // add text block acting as an interactive link
            pdfDocument.AddItems(ContentElement.FromMarkup("Visit our <a href='www.apitron.com'>website</a> " +
                "if you want to learn more about our PDF components."));

            string outputFileName = "document.pdf";

            // save the PDF file
            using (Stream outputStream = File.Create (outputFileName))
            {
                pdfDocument.Write (outputStream, resourceManager);
            }

            // open file with default system viewer
            Process.Start (outputFileName);
        }
    }
}
```

Pic. 2 Main program code file

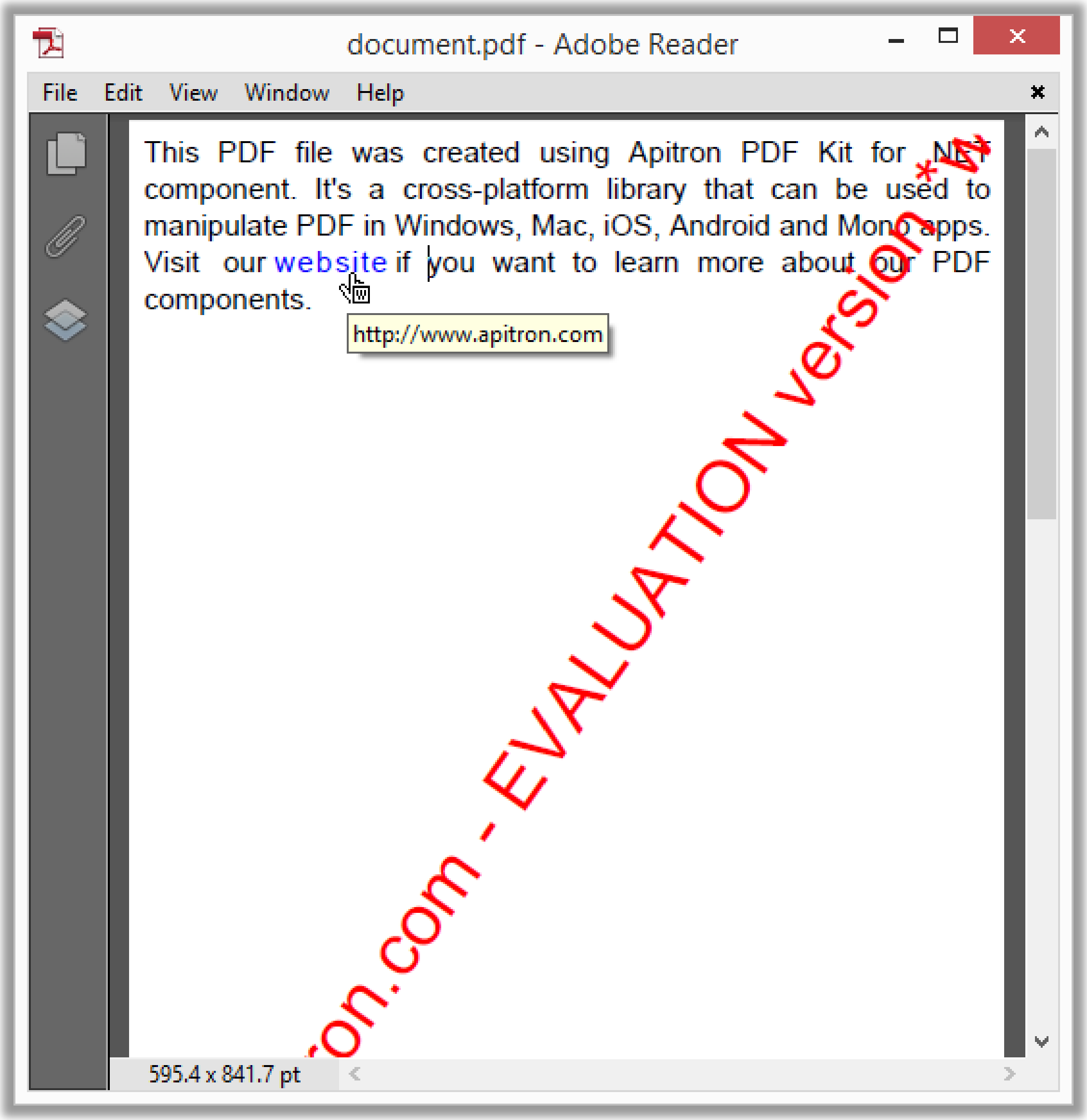
Writing the code

The code used in this sample is provided below; it creates a simple PDF file with a few text blocks. It shows how to use styles and markup parsing for creating several TextBlocks at once. The automatic link creation for TextBlocks is demonstrated as well.

```
using System;
using Apitron.PDF.Kit;
using Apitron.PDF.Kit.FlowLayout.Content;
using System.IO;
using Apitron.PDF.Kit.Styles;
using Apitron.PDF.Kit.Styles.Text;
using System.Diagnostics;
using Apitron.PDF.Kit.FixedLayout.Resources;
using Apitron.PDF.Kit.FixedLayout.Resources.Fonts;
using Font = Apitron.PDF.Kit.Styles.Text.Font;
using Apitron.PDF.Kit.Styles.Appearance;
using Apitron.PDF.Kit.FlowLayout;

namespace CreatePDFFileSample
{
    class MainClass
    {
        public static void Main (string[] args)
        {
            // create flow layout PDF document and resource manager
            ResourceManager resourceManager = new ResourceManager ();
            FlowDocument pdfDocument = new FlowDocument () {Margin = new Thickness(10), Align = Align.Justify};

            // register styles
            pdfDocument.StyleManager.RegisterStyle ("textblock",new Style()
                {
                    Font = new Font(StandardFonts.Helvetica, 20),
                    Color = RgbColors.Black,
                });

            pdfDocument.StyleManager.RegisterStyle ("textblock.a", new Style()
                {
                    Color = RgbColors.Blue
                });

            // add simple text blok element
            pdfDocument.Add(new TextBlock ("This PDF file was created using Apitron PDF Kit for .NET component. " +
                "It's a cross-platform library that can be used to manipulate PDF in Windows, " +
                "Mac, iOS, Android and Mono apps."));

            // add text containing the block acting as an interactive link
            pdfDocument.AddItems(ContentElement.FromMarkup("Visit our <a href='www.apitron.com'>website</a> " +
                "if you want to learn more about our PDF components."));

            string outputFileName = "document.pdf";

            // save the PDF file
            using (Stream outputStream = File.Create (outputFileName))
            {
                pdfDocument.Write (outputStream, resourceManager);
            }

            // open file with default system viewer
            Process.Start (outputFileName);
        }
    }
}
```

Results

If you run the sample created above, you’ll get the results shown on the images below:



This PDF file was created using Apitron PDF Kit for .NET component. It's a cross-platform library that can be used to manipulate PDF in Windows, Mac, iOS, Android and Mono apps. Visit our website if you want to learn more about our PDF components.

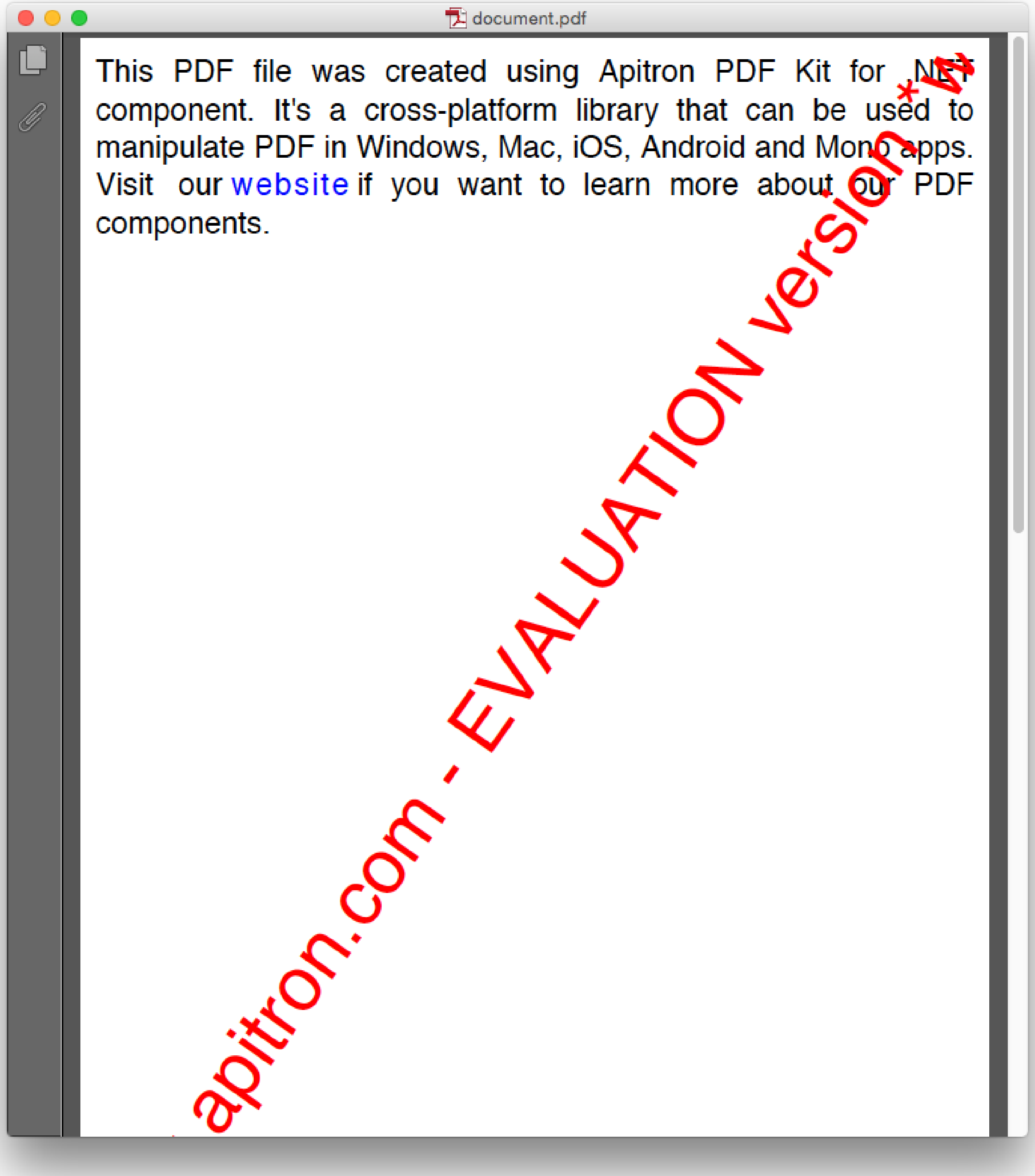
Pic. 3 Resulting PDF document (Windows)

You can see that the text is justified (because we set the document’s property Align), and it also contains a web link (pointed by the small hand cursor). If you click on this link, you’ll be navigated to the URL shown by the tooltip. The link in this document is represented by the Link Annotation object.

Running the app on OS X produces the same results. Just copy it to your OS X machine and execute the following command in terminal:

```
mono test/CreatePDFFileSample.exe
```

It assumes you have created a directory named “test” in your home folder and put files there. An image below shows the resulting PDF document:



This PDF file was created using Apitron PDF Kit for .NET component. It's a cross-platform library that can be used to manipulate PDF in Windows, Mac, iOS, Android and Mono apps. Visit our website if you want to learn more about our PDF components.

Pic. 4 Resulting PDF document (OS X)

Conclusion

Creation of the cross platform PDF processing application might be easier than you think with the right tools in hands. We've have shown how you could create a simple MONO based pdf processing app in this article. But the Apitron PDF Kit .NET component is not limited to such simple cases; it can be used to perform complex PDF processing tasks including: content manipulation and extraction, document merging, adding digital signatures, creation of PDF forms and much more, and provides a uniform API for all supported platforms.