Convert and view PDF documents in Xamarin Forms applications

Written by Apitron Documentation Team

Introduction

Xamarin.Forms offers you a flexible cross-platform alternative to create data entry applications targeting multiple platforms at once. Sometimes you might need to create PDF file based on entered data, or show the existing PDF document to user. In this article we'll demonstrate how to render existing PDF documents in a Xamarin.Forms app targeting iOS and Android using Apitron PDF Rasterizer component.

Form layout and code

We'll use very simple layout – a button to trigger rendering and an image to display the result. See the XAML and code behind below

```
<?xml version="1.0" encoding="UTF-8"?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml" x:Class="XamarinFormsSample.MyPage">
    <ContentPage.Content>
         <StackLayout Orientation="Vertical" Spacing="10" VerticalOptions="FillAndExpand"
          HorizontalOptions="FillAndExpand">
             <StackLayout Padding="0,10,0,10">
                <Button x:Name="btnRenderPDF" Clicked="OnRenderPdfClicked" Text="Render"/>
            </StackLayout>
            <ScrollView Orientation="Horizontal">
            <ScrollView Orientation="Vertical">
                 <Image x:Name="myImage" Aspect="Fill" VerticalOptions="StartAndExpand"
                 HorizontalOptions="StartAndExpand"/>
            </ScrollView>
            </ScrollView>
        </StackLayout>
    </ContentPage.Content>
</ContentPage>
```

```
public partial class MyPage : ContentPage
{
    public MyPage ()
    {
        InitializeComponent ();
    }

    void OnRenderPdfClicked(object sender, EventArgs args)
    {
        Assembly currentAssembly = typeof(MyPage).GetTypeInfo().Assembly;

        using (Stream resourceStream =
          currentAssembly.GetManifestResourceStream ("XamarinFormsSample.Data.testfile.pdf"))
        {
            byte[] buffer = new byte[resourceStream.Length];
            resourceStream.Read (buffer, 0, buffer.Length);

            var renderer = DependencyService.Get<IRenderer>();
            myImage.Source = ImageSource.FromStream (()=>
                {
                    return renderer.RenderToStream(buffer,0);
                });
        }
    }
}
```

We read the PDF document from resources and pass it to platform dependent renderer implementation requested using the dependency service. It returns an image stream used as image source.

Android implementation

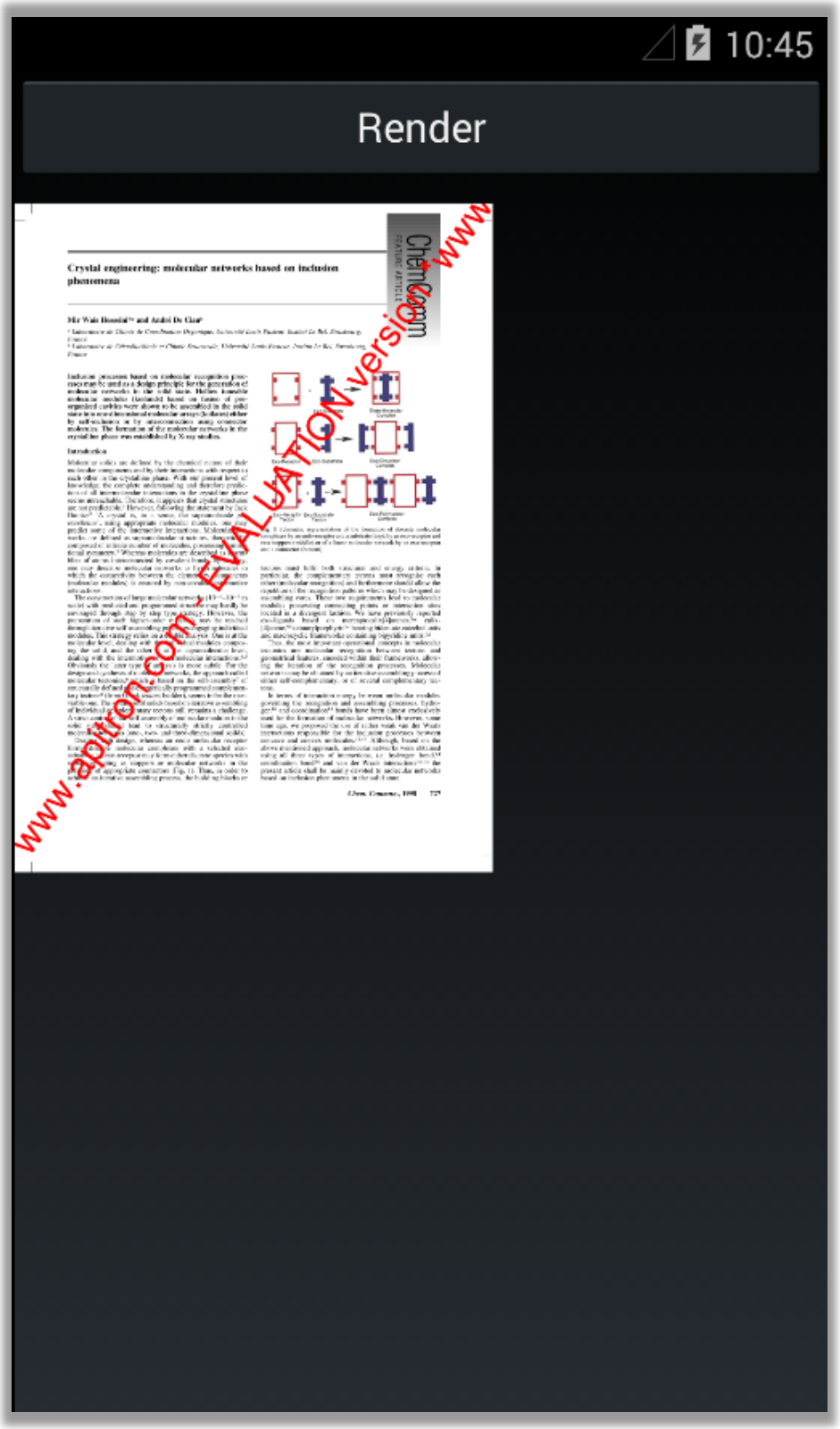
```
using System;
using Xamarin.Forms;
using XamarinFormsSample;
using Apitron.PDF.Rasterizer;
using Android.Graphics;
using System.IO;
using XamarinFormsSample.Droid;
using Java.Nio;

[assembly: Dependency(typeof(Renderer))]

namespace XamarinFormsSample.Droid
{
    /// <summary>
    /// Android specific implementation of <see cref="XamarinFormsSample.IRenderer"/> interface.
    /// </summary>
    public class Renderer:IRenderer
    {
        public Renderer ()
        {
        }

        #region IRenderer implementation
        public System.IO.Stream RenderToStream (byte[] documentData, int pageIndex)
        {
            using (MemoryStream ms = new MemoryStream (documentData))
            {
                // open document
                using (Document doc = new Document (ms))
                {
                    // prepare for rendering
                    int width = (int)doc.Pages [pageIndex].Width;
                    int height = (int)doc.Pages [pageIndex].Height;
                    // render as ints array
                    int[] renderedPage = doc.Pages [pageIndex].RenderAsInts (width, height,
                        new Apitron.PDF.Rasterizer.Configuration.RenderingSettings ());

                    // create bitmap and save it to stream
                    Bitmap bm = Bitmap.CreateBitmap (renderedPage, width, height,
                        Bitmap.Config.Argb8888);

                    MemoryStream outputStream = new MemoryStream ();
                    bm.Compress (Bitmap.CompressFormat.Png, 100, outputStream);

                    outputStream.Position = 0;

                    return outputStream;
                }
            }
        }
        #endregion
    }
}
```

iOS implementation

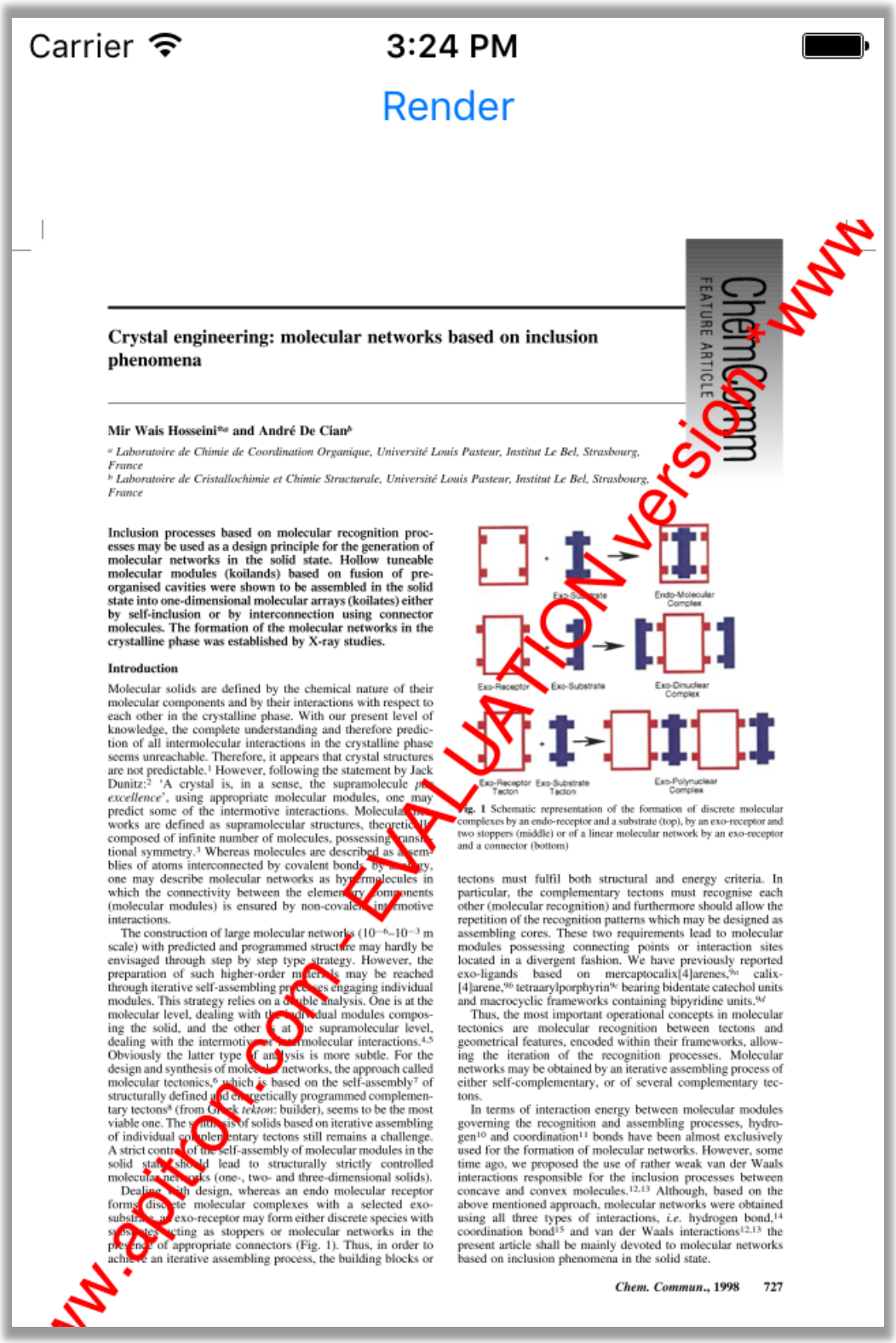
```
using …

[assembly: Dependency(typeof(Renderer))]

namespace XamarinFormsSample.iOS
{
    // iOS specific implementation of <see cref="XamarinFormsSample.IRenderer"/> interface.
    public class Renderer:IRenderer
    {
        public Renderer ()
        {
        }
        public System.IO.Stream RenderToStream (byte[] documentData, int pageIndex)
        {
            using (MemoryStream ms = new MemoryStream (documentData))
            {
                // open document
                using (Document doc = new Document (ms))
                {
                    // prepare for rendering
                    int width = (int)doc.Pages [pageIndex].Width;
                    int height = (int)doc.Pages [pageIndex].Height;
                    // render the page to a raw bitmap data represented by byte array
                    byte[] imageData =ConvertBGRAtoRGBA(doc.Pages[pageIndex].RenderAsBytes(width,
                        height, new RenderingSettings (), null));

                    // create CGDataProvider which will serve CGImage creation
                    CGDataProvider dataProvider =new CGDataProvider(imageData,0,imageData.Length);

                    // create core graphics image using data provider created above, note that
                    // we use CGImageAlphaInfo.Last(ARGB) pixel format
                    CGImage cgImage = new CGImage(width,height,8,32,width*4,
                        CGColorSpace.CreateDeviceRGB(),CGImageAlphaInfo.Last,dataProvider,null,
                        false, CGColorRenderingIntent.Default);
                    // create UIImage and save it to gallery
                    UIImage finalImage = new UIImage (cgImage);
                    return finalImage.AsPNG ().AsStream();
                }
            }
        }
        /// <summary>
        /// Converts the BGRA data to RGBA.
        /// </summary>
        /// <returns>Same byte array but with RGBA color dara.</returns>
        /// <param name="bgraData">Raw bitmap data in BGRA8888 format .</param>
        byte[] ConvertBGRAtoRGBA(byte[] bgraData)
        {
            // implemented simple conversion, swap 2 bytes.
            byte tmp;
            for(int i=0,k=2;i<bgraData.Length;i+=4,k+=4)
            {
                tmp = bgraData [i]; bgraData [i] = bgraData [k];bgraData [k] = tmp;
            }
            return bgraData;
        }
    }
}
```

Results

The source PDF file rendered on both platforms is shown on the images below:



Pic. 1 Render PDF in Xamarin.Forms App (Android)



Pic. 2 Render PDF in Xamarin.Forms app (iOS)

Conclusion

In this article we've shown how to convert to image and view PDF documents in Xamarin Forms applications. We used Apitron PDF Rasterizer .NET component for that, but if you need to create PDF documents or manipulate them you can use Apitron PDF Kit and do whatever you want with PDF files. Text extraction, context generation, adding signatures and many other features are wrapped by the easy to use API for your convenience. Contact us if you need any help and we'll be happy to assist you.