Extracting pages from a PDF document and saving them as separate image files, C# edition

devblogs.microsoft.com/oldnewthing/20170626-00

June 26, 2017

Raymond Chen

Today's Little Program extracts the pages from a PDF document and saves them as separate image files. Why? Because I needed to do that.

I'll start with the PDF Document sample program and change it so that instead of displaying pages on the screen, it saves them to disk.

Take the C# sample and make these changes to `Scenario1_Render.xaml.cs`:

```
private async void ViewPage()
{
    rootPage.NotifyUser("", NotifyType.StatusMessage);

    uint pageNumber;
    if (!uint.TryParse(PageNumberBox.Text, out pageNumber) ||
        (pageNumber < 1) || (pageNumber > pdfDocument.PageCount))
    {
        rootPage.NotifyUser("Invalid page number.", NotifyType.ErrorMessage);
        return;
    }

    // New: Ask the user for the output file.
    var save = new FileSavePicker();
    save.FileTypeChoices["PNG image"] = new[] { ".png" };
    var outfile = await save.PickSaveFileAsync();
    if (outfile == null) return;

    Output.Source = null;
    ProgressControl.Visibility = Visibility.Visible;

    // Convert from 1-based page number to 0-based page index.
    uint pageIndex = pageNumber - 1;

    using (PdfPage page = pdfDocument.GetPage(pageIndex))
    using (var transaction = await outfile.OpenTransactedWriteAsync())
    {
        await page.RenderToStreamAsync(transaction.Stream);
    }
    ProgressControl.Visibility = Visibility.Collapsed;
}
```

Actually, I kind of gutted the program and replaced it with my own stuff. The only interesting parts that remain from the original program are the `LoadDocument` method (not shown here) which loads the PDF file into the `pdfDocument` variable, and the part that obtains the `PdfPage` from the user-specified page number.

We ask for the output file, obtain a write stream to that file, and ask the `page` to render into that stream. The default options generate a bitmap in PNG format whose size matches the declared `Size` of the page.

The PNG format was fine for my purposes, but the size wasn't. WinRT view pixels are 1/96 of an inch, so the resulting bitmap was rendered as if printed to a 96 DPI printer. That's the resolution of a first-generation fax machine, which isn't all that great. I wanted 192 DPI, so I needed to render the image at double-size.

```
    using (PdfPage page = pdfDocument.GetPage(pageIndex))
    using (var transaction = await outfile.OpenTransactedWriteAsync())
    {
        var options = new PdfPageRenderOptions();
        options.DestinationHeight = (uint)(page.Size.Height * 2);
        options.DestinationWidth = (uint)(page.Size.Width * 2);
        await page.RenderToStreamAsync(transaction.Stream, options);
    }
```

(If I had wanted to change the file format, I'd have set the `options.BitmapEncoderId` to something like `BitmapEncoder.JpegEncoderId`.)

I didn't have a large document to convert, so changing the page number and clicking the (now-mislabeled) "View" button a dozen times wasn't that big of a deal.

For the rest of the week, I'm going to be translating this program into C++/CX (twice) and JavaScript (twice).

Twice?

Yes, twice. You'll see.

And then there will be a bonus.

I can sense your anticipation.

Raymond Chen

Follow