Windows 8 开发 31 日

第 20 日

打印

译者：BeyondVincent(破船)

时间：2013.4.25

版本： 2.0

关于破船

程序猿砌墙于云南昆明!

长期扎根移动软件开发!

爱跑步爱打篮球爱运动!

命中无大富大贵之面相!

愿健康与平淡相随一生!

你可以发邮件与破船取得联系: BeyondVincent@gmail.com

还可以关注破船的微博: 腾讯微博和新浪微博。

这里是破船的个人博客，欢迎光临：破船之家

关于 Windows 8 开发 31 日翻译

Windows 8 开发 31 日是由 Jeff Blankenburg 和 Clark Sell 原创的。

官方站点：http://31daysofwindows8.com/

涉及到两个版本：

XAML/C#（由 Jeff Blankenburg 撰写）

HTML5/JS （由 Clark Sell 撰写）

其中涉及到的资源和相关代码请到这里下载：

https://github.com/csell5/31DaysOfWindows8

在这里，由于破船对 HTML5/JS 不熟悉，所以只翻译 XAML/C#相关主题。

建议大家前往看原创内容，如果看不明白，再来这里看我翻译的相关内容。

如果翻译不正确的地方，可以通过上面的联系方式告诉破船。

破船祝你阅读愉快！

目录

第 20 日 打印

1.0. 介绍

今天，我将介绍来自你的 Windows 8 程序中的打印。在以前我作为一面软件开发人员时，唯一一次关注过打印是与一个 web page 相关，只需要调用 window.print()就会发生下面一些事情：

- 一个带可选项的打印对话框显示给用户
- 打印预览创建
- 基于页面的实际长度进行分页

尽管我已经使用 XAML 多年了，但是打印我并没有真正接触过。当接触之后，我感到惊讶的是：打印一个简单的页面需要很多的代码，不过我也知道了为什么会有这么多代码了。我们不得不创建这些代码，并且理由都很充分。

我们页面的布局并不能用在打印中纸张的大小和形状。

因此，我们需要提供一个我们页面的可以打印版本。今天的例子中，我使用一个简单的页面，里面有一些 RickTextBlock 控件，当然，你自己的页面可以根据

自己的需求做得更加复杂。

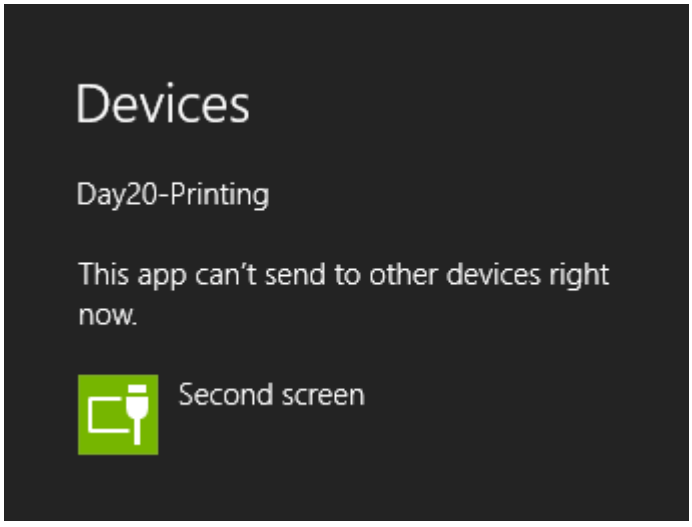
另外，在写这篇文章时，让我真的感觉到在 Windows 8 使用 XAML/C#来打印非常的困难。不仅需要处理生成打印预览（必须做）的所有步骤，还必须手动处理分页。这在 HTML5/JS 上是很方便的。确实让我难受了一把。

1.1. 让程序可以使用打印功能

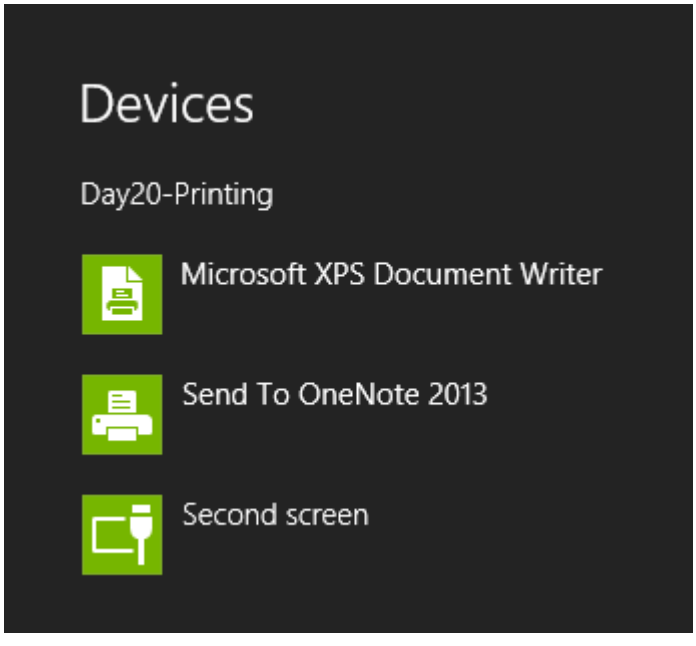
如果你尝试打印，你会发现基本的 Windows 8 程序还不可以使用。我们首先必须在页面中创建一个 PrintManager 对象。并且这不能再应用程序级别创建，而是在每一个需要打印的页面创建，所以也会引起代码的重复。（再次，我将 PrintManager 的创建代码放在 OnNavigatedTo 方法中，而在 OnNavigatingFrom 方法中取消注册相关事件，当然，如果你愿意的话，你也可以将这部分代码放在某个单击事件中）

```
PrintManager manager = PrintManager.GetForCurrentView();
manager.PrintTaskRequested += manager_PrintTaskRequested;
```

在页面中添加这段代码后（包括 PrintTaskRequested 注册），将会使用户的 device charm 菜单从下面图：



到下图：



我在写本文的时候，我的周围一台可用的打印机，不过选择打印到 OneNote 或 Microsoft XPS Document Writer 时的打印选项是一样的。用这种打印方式是很好

的测试方法。不仅还可以节省纸张，打印出来的电子文档看起来跟在打印机中打印出来的是一样的。

1.2. 实现打印

如果你只添加上面的两行代码，就尝试打印，那么会令你失望的。首先，你不能运行程序，我们还没有创建 event handler。其次，如果你创建了 event handler，当你使用打印机的时候，会得到一个错误。

首先，我们先定义一些变量：

First, we have a couple of global variables we'll be using:

```
PrintDocument document = null;
IPrintDocumentSource source = null;
List<UIElement> pages = null;
FrameworkElement page1;
protectedeventEventHandler pagesCreated;
protectedconstdouble left = 0.075;
protectedconstdouble top = 0.03;
```

下面我逐行来解释一下，上面这些代码：

- PrintDocument—这是我们将要添加内容的一个实际文档。
- IPrintDocumentSource—这是 PrintDocument 对象的 source。
- List<UIRlement>—这个是存储我们已经分好页的,即将打印的 document 页面。
- FrameWorkElement—这会保存这 document 的第一个 page。

■ Protected event—这将用于创建任何一个 page 时的事件。

■ left，top—这将用于设置页面的实际边距。

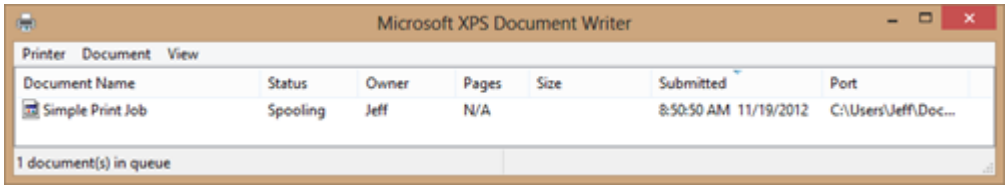
先来看看 OnNavigatedTo 方法，这是所有处理的开始：

```
protectedoverridevoid OnNavigatedTo(NavigationEventArgs e)
{
        document = newPrintDocument();
        source = document.DocumentSource;

        document.Paginate += printDocument_Paginate;
        document.GetPreviewPage += printDocument_GetPreviewPage;
        document.AddPages += printDocument_AddPages;

PrintManager manager = PrintManager.GetForCurrentView();
        manager.PrintTaskRequested += manager_PrintTaskRequested;

        pages = newList<UIElement>();

        PrepareContent();
}
```

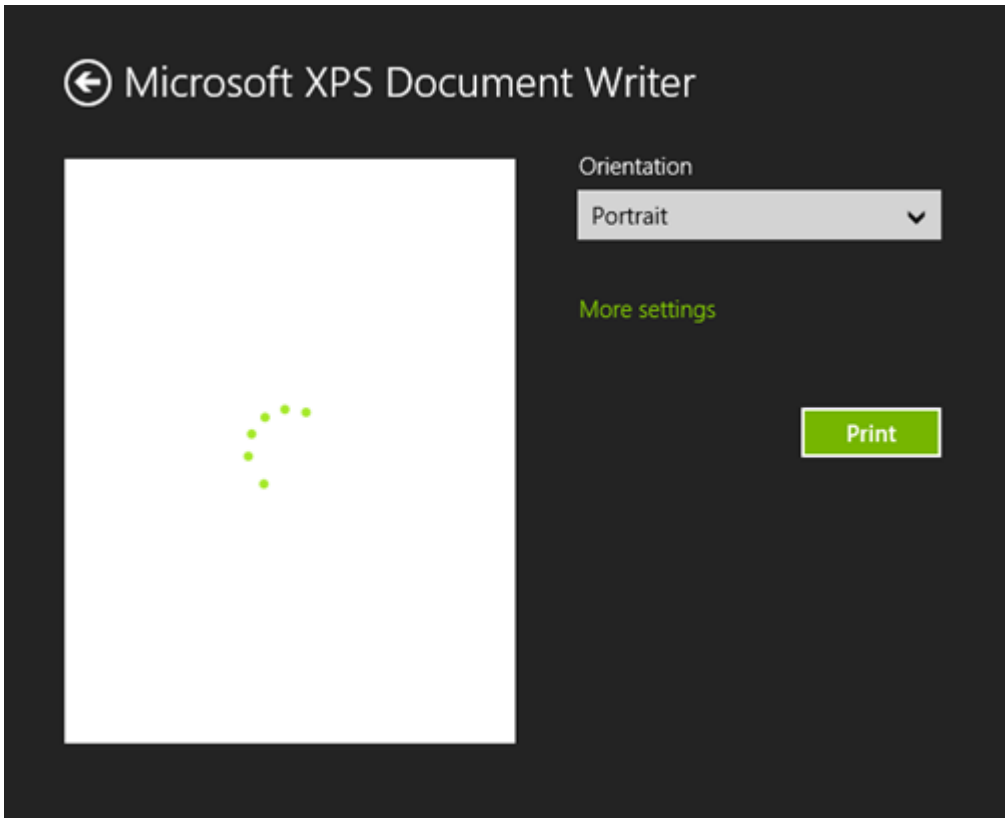
首先，我实例化了一个 document 和 source 对象，然后给 document 创建了 3 个新的 event handler：Paginate、GetPreviewPage 和 AddPages。最后，我创建了一个 PrintManager（之前的两行代码有提到过），最后两步，我创建了一个 pages List<> ,并调用了方法 PrepareContent()。之后我们会再看到 PrepareContent 方法的。现在我们先来看看下面这个 event handler。

```
void manager_PrintTaskRequested(PrintManager sender, PrintTaskRequestedEventArgs args)
{
PrintTask task = null;
        task = args.Request.CreatePrintTask("Day #20 - Simple Print Job", sourceRequested =>
        {
                sourceRequested.SetSource(source);
        });
}
```

当用户选择一个打印机时，上面的这个方法会被调用。在里面，我创建了一个 PrintTast 对象，并设置一个名字，以及之前创建好的 source。名字将出现在打印队列中，如下：



一旦选择了打印机，将会看到一个新的窗口，如下：



你会注意到两个东西：

- 我们还没有指定打印什么内容。

- 打印预览好像要显示什么东西。

我们先来搞定第一个问题——内容填充。下面是之前提到过的 PrepareContent() 方法：

```
privatevoid PrepareContent()
{
if (page1 == null)
      {
            page1 = newPageForPrinting();
StackPanel header = (StackPanel)page1.FindName("header");
            header.Visibility = Windows.UI.Xaml.Visibility.Visible;
      }

      PrintContainer.Children.Add(page1);
      PrintContainer.InvalidateMeasure();
      PrintContainer.UpdateLayout();
}
```

在这个方法中，有几个重要的事情会发生。首先我创建了一个新的 PageForPrinting 对象给 page1，PageForPrinting 的任务是将程序中的一个 XAML 文件格式化为打印的 page。其次，我们进入到 PageForPrinting，查找一个名为“header”的 element（这完全是一个可选的，但它是一个非常有用的演示。），这是一个 StackPanel，在格式化的 page 中，只有打印的时候才显示出来。这个方法可以用来添加页眉和页脚，特别是当你不想让页眉和页脚显示在屏幕中时。最后，你可以看到我将我将 page1 添加到名为 PrintContainer 的 element 中。目的是将我们的 page 提交给已存在的可视化树中，以强制的进行 layout，以便所有的 containers 都可以在里面发布内容。

下面是 PageFroPrinting.xaml 文件：

```
<Page
```

```
 x:Class="Day20_Printing.PageForPrinting"
 xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
 xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
 xmlns:local="using:Day20_Printing"
 xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
 xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
 mc:Ignorable="d">

<Grid x:Name="printableArea">

<Grid.RowDefinitions>
<RowDefinition/>
<RowDefinition Height="3*"/>
<RowDefinition Height="3*"/>
<RowDefinition Height="4*"/>
<RowDefinition Height="Auto"/>
</Grid.RowDefinitions>
<Grid.ColumnDefinitions>
<ColumnDefinition Width="6*" />
<ColumnDefinition Width="4*"/>
</Grid.ColumnDefinitions>

<StackPanel x:Name="header" Grid.Row="0" Grid.ColumnSpan="2" Height="60" Visibility="Collapsed">
<StackPanel Orientation="Horizontal" >
<Image Source="ms-appx:///Assets/StoreLogo.png" HorizontalAlignment="Left" Stretch="None"/>
<RichTextBlock Foreground="Black" FontSize="20" TextAlignment="Left" FontFamily="Segoe UI">
<Paragraph>Day #20 - Printing</Paragraph>
</RichTextBlock>
</StackPanel>
</StackPanel>

<RichTextBlock Foreground="Black" x:Name="textContent" FontSize="18" Grid.Row="1" Grid.ColumnSpan="2"
OverflowContentTarget="{Binding ElementName=firstLinkedContainer}"
 IsTextSelectionEnabled="True" TextAlignment="Left" FontFamily="Segoe UI" VerticalAlignment="Top"
HorizontalAlignment="Left">
<Paragraph FontSize="32">Lorem ipsum dolor sit amet, consectetur</Paragraph>
<Paragraph></Paragraph>
<Paragraph>Sed convallis ornare velit et interdum. Donec sapien neque, aliquet consequat convallis at, interdum et
enim. Donec iaculis, lectus vel pulvinar cursus, lectus diam interdum ante, a rhoncus tortor quam porta metus. Class
aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Etiam pulvinar fringilla
vestibulum. Pellentesque pharetra nunc in turpis tempus sed faucibus ligula sagittis. Praesent hendrerit est vitae lorem
mattis in porttitor urna vestibulum. Phasellus adipiscing aliquam libero ac adipiscing. In a erat sit amet erat
sollicitudin bibendum id vitae dui. Vestibulum non consequat nisl. Vestibulum ante ipsum primis in faucibus orci
luctus et ultrices posuere cubilia Curae; Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac
turpis egestas. Mauris elit nisi, blandit et porttitor quis, malesuada nec mi.</Paragraph>
<Paragraph></Paragraph>
<Paragraph>Aliquam erat volutpat. In non urna ut libero ultricies fringilla. Proin tellus neque, aliquam lacinia
consequat at, vulputate et arcu. Maecenas odio nunc, lobortis sit amet pulvinar sit amet, accumsan et leo. Suspendisse
erat lectus, commodo ac auctor eget, rutrum in mi. Suspendisse potenti. Proin ac elit non lacus rutrum mollis.
Vivamus venenatis, tellus vel placerat lacinia, arcu ligula dignissim orci, consectetur consectetur eros massa vel nulla.
Quisque malesuada iaculis ornare. Nullam tincidunt accumsan egestas. Mauris sit amet scelerisque arcu. Proin
```

```
euismod sodales magna faucibus commodo. Nam in fringilla orci. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos.</Paragraph>
<Paragraph></Paragraph>
<Paragraph>Sed eget nunc quis tellus interdum aliquet. Suspendisse rhoncus malesuada nisi a imperdiet. Suspendisse ullamcorper mi sed purus tristique interdum. Mauris lobortis, ante ultrices varius consequat, eros ante hendrerit enim, vulputate convallis dui ligula eget velit. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Donec eget lectus fermentum nisi consequat dictum. Sed volutpat justo non purus semper vel pretium enim molestie. Nam consectetur, lectus quis feugiat malesuada, neque nunc faucibus velit, nec vehicula risus est id sapien. Vestibulum ut metus massa, ut placerat lacus. Fusce condimentum vehicula tortor, nec vestibulum ligula iaculis ut. Nulla facilisi. Phasellus tincidunt scelerisque erat, ut fermentum urna pretium eu. Donec ut nibh orci. Curabitur sodales metus dictum mauris varius vitae mollis tellus pulvinar. Quisque facilisis ligula sed risus laoreet non lacinia odio luctus. Nam lobortis rhoncus felis vitae ultrices.</Paragraph>
<Paragraph></Paragraph>
<Paragraph>Aliquam erat volutpat. In non urna ut libero ultricies fringilla. Proin tellus neque, aliquam lacinia consequat at, vulputate et arcu. Maecenas odio nunc, lobortis sit amet pulvinar sit amet, accumsan et leo. Suspendisse erat lectus, commodo ac auctor eget, rutrum in mi. Suspendisse potenti. Proin ac elit non lacus rutrum mollis. Vivamus venenatis, tellus vel placerat lacinia, arcu ligula dignissim orci, consectetur consectetur eros massa vel nulla. Quisque malesuada iaculis ornare. Nullam tincidunt accumsan egestas. Mauris sit amet scelerisque arcu. Proin euismod sodales magna faucibus commodo. Nam in fringilla orci. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos.</Paragraph>
<Paragraph></Paragraph>
<Paragraph>Sed eget nunc quis tellus interdum aliquet. Suspendisse rhoncus malesuada nisi a imperdiet. Suspendisse ullamcorper mi sed purus tristique interdum. Mauris lobortis, ante ultrices varius consequat, eros ante hendrerit enim, vulputate convallis dui ligula eget velit. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Donec eget lectus fermentum nisi consequat dictum. Sed volutpat justo non purus semper vel pretium enim molestie. Nam consectetur, lectus quis feugiat malesuada, neque nunc faucibus velit, nec vehicula risus est id sapien. Vestibulum ut metus massa, ut placerat lacus. Fusce condimentum vehicula tortor, nec vestibulum ligula iaculis ut. Nulla facilisi. Phasellus tincidunt scelerisque erat, ut fermentum urna pretium eu. Donec ut nibh orci. Curabitur sodales metus dictum mauris varius vitae mollis tellus pulvinar. Quisque facilisis ligula sed risus laoreet non lacinia odio luctus. Nam lobortis rhoncus felis vitae ultrices.</Paragraph>
</RichTextBlock>
<RichTextBlockOverflow x:Name="firstLinkedContainer" OverflowContentTarget="{Binding ElementName=continuationPageLinkedContainer}" Grid.Row="2" Grid.Column="0"/>
<RichTextBlockOverflow x:Name="continuationPageLinkedContainer" Grid.Row="3" Grid.ColumnSpan="2"/>
<Image Source="ms-appx:///Assets/Logo.png" x:Name="scenarioImage" HorizontalAlignment="Right" Grid.Row="2" Grid.Column="1" Margin="10"/>

<StackPanel x:Name="footer" Grid.Row="4" Grid.Column="0" VerticalAlignment="Top" Visibility="Collapsed">
<Image Source="ms-appx:///Assets/StoreLogo.png" HorizontalAlignment="Left" Stretch="None"/>
<RichTextBlock Foreground="Black" FontSize="16" TextAlignment="Left" FontFamily="Segoe UI">
<Paragraph>Copyright © 31 Days of Windows 8.  But please reuse this.</Paragraph>
</RichTextBlock>
</StackPanel>
</Grid>
</Page>
```

在这里，使用 RichTextBlockOverflow 衔接换页。这个 page 的 behind code 没有任何改变，如果要想董岱的加载这些内容时，那么就需要在 behind code 里面处

理了。

在上面，我们处理了用户选择打印机，以及格式化好了需要打印 page。下一步是处理之前创建的 3 个 event handler，首先从 Paginate：

```
void printDocument_Paginate(object sender, PaginateEventArgs e)
{
    pages.Clear();
    PrintContainer.Children.Clear();

RichTextBlockOverflow lastRTBOOnPage;
PrintTaskOptions printingOptions = ((PrintTaskOptions)e.PrintTaskOptions);
PrintPageDescription pageDescription = printingOptions.GetPageDescription(0);

    lastRTBOOnPage = AddOnePrintPreviewPage(null, pageDescription);

while (lastRTBOOnPage.HasOverflowContent && lastRTBOOnPage.Visibility ==
Windows.UI.Xaml.Visibility.Visible)
    {
        lastRTBOOnPage = AddOnePrintPreviewPage(lastRTBOOnPage, pageDescription);
    }

if (pagesCreated != null)
    {
        pagesCreated.Invoke(pages, null);
    }

PrintDocument printDoc = (PrintDocument)sender;

    printDoc.SetPreviewPageCount(pages.Count, PreviewPageCountType.Intermediate);
}
```

每次这个方法被调用的时候，我们都要清楚我们的“pages”列表，以及在 MainPage 中提交给 PrintContainer 的任何内容。(只是将 canvas 的 Opacity 设置为 0，这样我们可以在里面添加 content，但是用户看不到。)

我们还创建了一个 RichTextBlockOverflow 对象，用来跟踪当前处于哪个 page。最后，调用了 AddOnePrintPreviewPage 方法，这个方法为我们做了很多事情，计算 page 的 size，设置 margins，然后将 page 保存到 pages 列表中。还可以看到，页

脚也被激活可见了。

```
privateRichTextBlockOverflow AddOnePrintPreviewPage(RichTextBlockOverflow lastRTBOAdded,
PrintPageDescription printPageDescription)
{
FrameworkElement page;
RichTextBlockOverflow link;

if (lastRTBOAdded == null)
        {
                page = page1;
StackPanel footer = (StackPanel)page.FindName("footer");
                footer.Visibility = Windows.UI.Xaml.Visibility.Collapsed;
        }
else
        {
                page = newContinuationPage(lastRTBOAdded);
        }

        page.Width = printPageDescription.PageSize.Width;
        page.Height = printPageDescription.PageSize.Height;

Grid printableArea = (Grid)page.FindName("printableArea");

double marginWidth = Math.Max(printPageDescription.PageSize.Width -
printPageDescription.ImageableRect.Width, printPageDescription.PageSize.Width * left * 2);
double marginHeight = Math.Max(printPageDescription.PageSize.Height -
printPageDescription.ImageableRect.Height, printPageDescription.PageSize.Height * top * 2);

        printableArea.Width = page1.Width - marginWidth;
        printableArea.Height = page1.Height - marginHeight;

        PrintContainer.Children.Add(page);
        PrintContainer.InvalidateMeasure();
        PrintContainer.UpdateLayout();

// Find the last text container and see if the content is overflowing
        link = (RichTextBlockOverflow)page.FindName("continuationPageLinkedContainer");

// Check if this is the last page
if (!link.HasOverflowContent && link.Visibility == Windows.UI.Xaml.Visibility.Visible)
        {
StackPanel footer = (StackPanel)page.FindName("footer");
                footer.Visibility = Windows.UI.Xaml.Visibility.Visible;
        }

// Add the page to the page preview collection
        pages.Add(page);
```

```
return link;
}
```

如果发现了 RichTextBlockOverflow（由我们调用的方法返回），我们还将循环处理这个过程，直到所有的 pages 都添加到列表中。

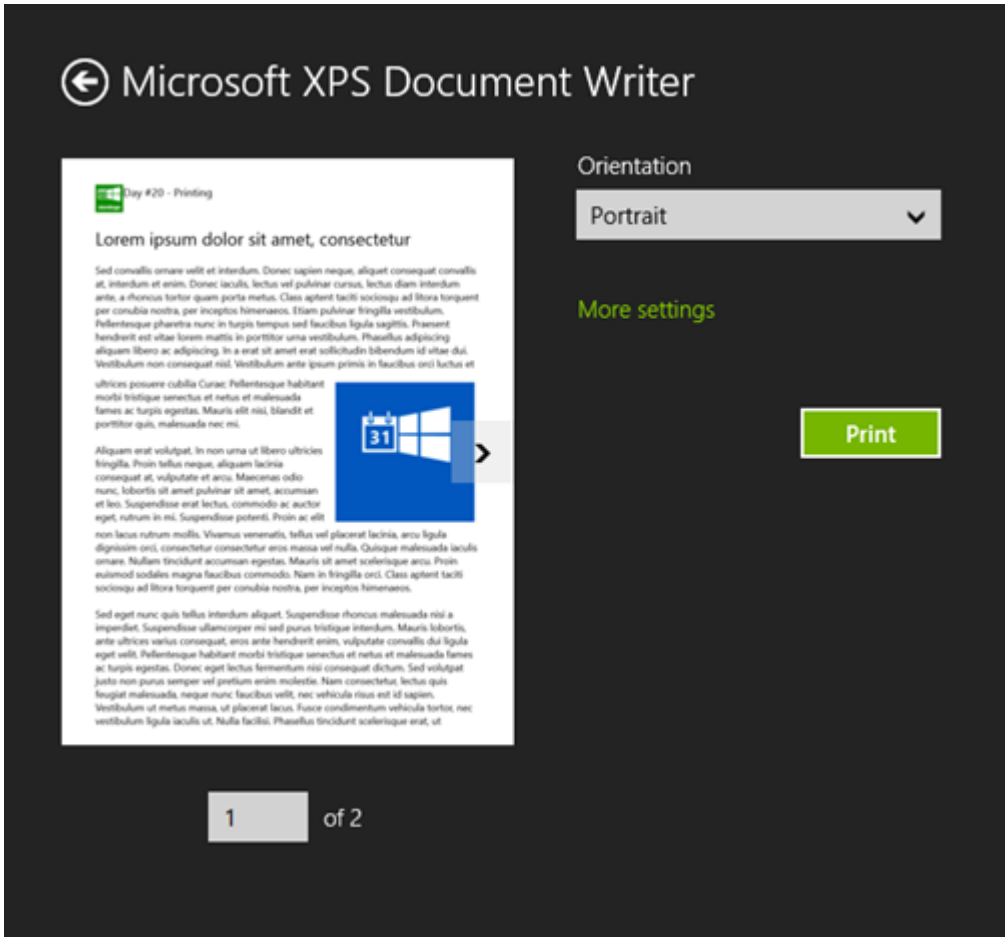
下一步，我们需要将这些 pages 添加到最初的 PrintDocument 对象中，在刚开始的时候，我们注册了一个 AddPages event。这个 event 是简单的循环处理在 Paginate 方法中创建的所有 pages。

```
void printDocument_AddPages(object sender, AddPagesEventArgs e)
{
for (int i = 0; i < pages.Count; i++)
      {
            document.AddPage(pages[i]);
      }

PrintDocument printDoc = (PrintDocument)sender;
      printDoc.AddPagesComplete();
}
```

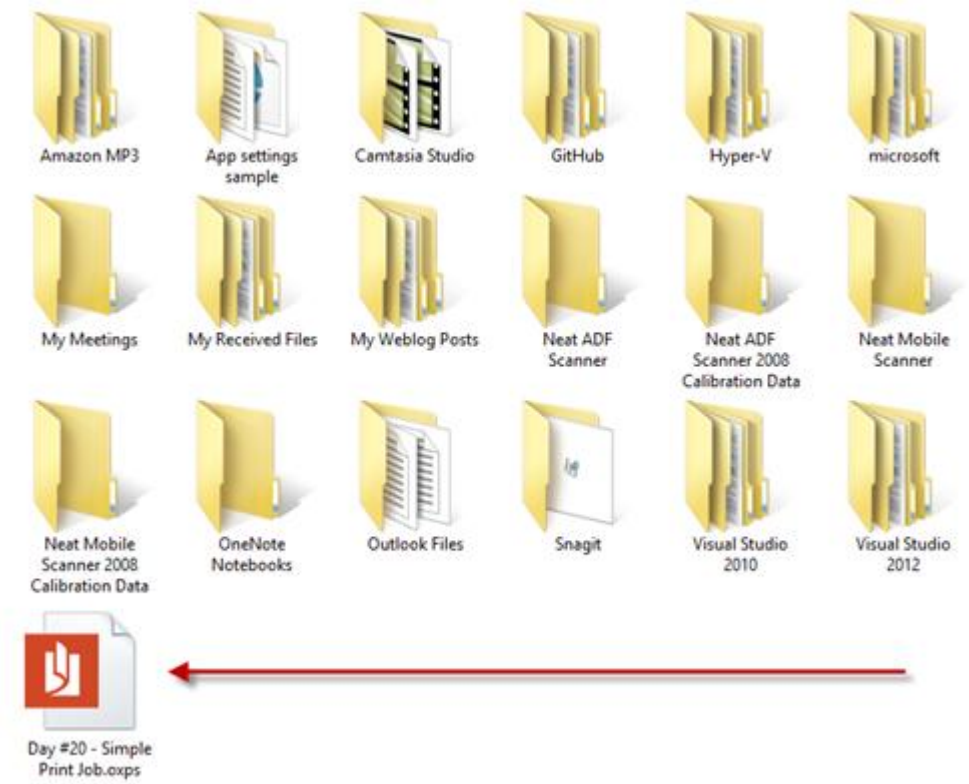
当打印预览对话框加载的时候，GetPreviewPage 会被触发调用。在这里，我们可以将第一页设置到打印预览窗口中：

```
void printDocument_GetPreviewPage(object sender, GetPreviewPageEventArgs e)
{
PrintDocument printDoc = (PrintDocument)sender;

      printDoc.SetPreviewPage(e.PageNumber, pages[e.PageNumber - 1]);
}
```

这是最后的一步。当提供了我们的预览页面，就做完了。在打印的时候，会看到如下的画面：



当 Microsoft XPS Document Writer，如果用户点击 Print，最终，会有一个文件生成到用户的 Documents 文件夹中：



打开那个文件，可以看到我们格式化的内容被分在两个 pages 中。

1.3. 总结

简短的说，这真的有点混乱，处理起来有点困难。我想要是有简单的方法就好了。以后，我会持续跟踪这个问题，并创建一个 PrintHelper 类，帮助你创建一个 page，并提供几种格式化，你只需要调用一个方法，所有的事情都搞定。如果打印像本文这么复杂的话，估计很少人会用。

点击下图，下载本文示例代码：

明天，我将介绍更多有用的主题（更容易实现）：从照相机捕获数据。到时候见！

感谢你的阅读！

如果对这篇文章有什么想法想法，可以与破船联系，破船的联系方式在文章开头。

破船