PdfViewer for UWP

2018.03.07 更新

グレープシティ株式会社

PDFViewer for UWP

目次

PdfViewer for UWP 2
主な特長 3
クイックスタート 4
手順1:アプリケーションへの C1PdfViewer の追加 4-5
手順2:C1PdfViewer アプリケーションへのコードの追加 5-7
手順3:C1PdfViewer アプリケーションの実行 7
C1PdfViewer の使い方 8
ドキュメントの読み込み 8
非同期読み込み 8
暗号化されたファイルの読み込み 8
タッチ操作 9
ビューモード 9
方向 9
タスク別ヘルプ 10
アプリケーションリソースからドキュメントを読み込む 10-11
Web からドキュメントを読み込む 11
C1PdfDocument で作成された PDF ファイルを読み込む 11-12
保護されている可能性のあるファイルを開く 12-14

PdfViewer for UWP

ユニバーサルWindowsアプリにドキュメント表示機能を追加します。**PdfViewer for UWP** は、外部アプリケーションがなくても、アプリケーション内で単純な PDF ドキュメントを表示できるようにします。任意の PDF ドキュメントをロードすることができます。

主な特長

PdfViewer for UWP には、次の主要な機能があります。

- **PDF ファイルのロードと表示**
  C1PdfViewer コントロールを使用して、ユニバーサルWindowsアプリアプリケーションで PDF ファイルをロードして表示します。この XAML コントロールは、デスクトップや外部の Adobe 製品に依存することなくファイルを表示または保存します。コンテンツは、ネイティブな UWP 要素として解析されてレンダリングされます。
- **マルチタッチジェスチャーのサポート**
  ユーザーは、ページをスライドしてスクロールできるほか、ドキュメントをピンチしてズームすることができます。ズームすることで、狭い画面にコンテンツを読みやすく表示できます。
- **水平方向**
  C1PdfViewer コントロールは、垂直方向と水平方向の両方をサポートします。それには、**Orientation** プロパティを設定するだけです。
- **暗号化されたファイルのサポート**
  C1PdfViewer コントロールは、暗号化されたファイルの表示をサポートします。**C1PdfViewer.LoadDocument** メソッドには、暗号化されたファイルを表示するためのオプションのパスワードパラメータがあります。
- **PDF 仕様のサポート**
  C1PdfViewer は、PDF 1.5 仕様のサブセットをサポートしています。ただし、暗号化、特殊フォント、あまり使用されない画像形式などの重要な制限がいくつかあります。サポートされていないコンテンツを含むドキュメントもレンダリングされますが、正しく書式設定されない場合があります。目的の PDF ファイルで使用される機能を制御環境で事前にテストできる場合は、C1PdfViewer を使用することをお勧めします。すべての制限については、マニュアルで確認することができます。
- **PDF からのページの取得**
  読み込んだ PDF のページを一連の **FrameworkElement** として取得し、各ページの表示方法をカスタマイズすることができます。これにより、既存の PDF ドキュメントを極めて柔軟に操作することができます。それには、**GetPages** メソッドを呼び出すだけです。GetPages メソッドの使用方法の詳細については、『PDFViewer での印刷』チュートリアルを参照してください。

クイックスタート

このクイックスタートガイドは、**PdfViewer for UWP** を初めて使用するユーザーのために用意されています。このクイックスタートガイドでは、最初に Visual Studio で新しいプロジェクトを作成し、アプリケーションに **PdfViewer for UWP** コントロールを追加し、コンテンツをコントロールに追加します。

手順1:アプリケーションへの C1PdfViewer の追加

この手順では、最初に Visual Studio で PdfViewer for UWP を使用する UWP スタイルのアプリケーションを作成します。プロジェクトをセットアップし、**C1PdfViewer** コントロールをアプリケーションに追加するには、次の手順に従います。

1. Visual Studio 2015 で、**[ファイル]**→**[新規作成]**→**[プロジェクト]**を選択します。
2. **[新しいプロジェクト]**ダイアログボックスで、**[テンプレート]**→**[Visual C#]**→**[Windows]**→**[ユニバーサル]**を選択します。テンプレートリストから、**[空白のアプリ(ユニバーサルWindows)]**を選択します。**[名前]**を入力し、**[OK]**をクリックしてプロジェクトを作成します。
3. MainPage.xaml が開いていない場合は開きます。<Grid> タグと </Grid> タグの間にカーソルを置き、1回クリックします。
4. 次の列定義と行定義を <Grid> タグと </Grid> タグの間に追加します。

▶ XAMLでマークアップを書く場合

マークアップ

```
<Grid.RowDefinitions>
<RowDefinition Height="Auto"/>
<RowDefinition/>
</Grid.RowDefinitions>
<Grid.ColumnDefinitions>
<ColumnDefinition/>
<ColumnDefinition Width="Auto"/>
</Grid.ColumnDefinitions>
```

グリッド内の要素が配置されて表示されます。

5. ツールボックスに移動し、[C1PdfViewer]アイコンをダブルクリックして、コントロールをアプリケーションに追加します。
6. C1PdfViewer のマークアップを次のように編集します。

▶ XAMLでマークアップを書く場合

マークアップ

```
<PdfViewer:C1PdfViewer x:Name="pdfViewer"
ViewMode="FitWidth"Grid.Row="1"Grid.ColumnSpan="2"/>
```

このマークアップは、コントロールに名前を付け、コントロールに PDF の幅全体が表示されるように ViewMode を設定し、コントロールのレイアウトをカスタマイズします。

7. ツールボックスに移動し、StackPanel アイコンをダブルクリックしてページに追加します。StackPanel のマークアップを次のように編集します。

▶ XAMLでマークアップを書く場合

マークアップ

```
<StackPanel Orientation="Horizontal" Margin="8" VerticalAlignment="Center">
<TextBlock Text="{Binding ElementName=pdfViewer, Path=PageNumber}" FontSize="20"
```

```
      Foreground="{StaticResourceApplicationForegroundThemeBrush}" />
<TextBlock Text=" / " Foreground="{StaticResource
ApplicationForegroundThemeBrush}"
      FontSize="20"/>
<TextBlock Text="{Binding ElementName=pdfViewer, Path=PageCount}" FontSize="20"
      Foreground="{StaticResourceApplicationForegroundThemeBrush}" />
</StackPanel>
```

このマークアップは、StackPanel に3つの TextBlock コントロールを追加します。

8. StackPanel の終了タグアイコンの直後に次のマークアップを追加して、ページに Button を追加します。

▶ XAMLでマークアップを書く場合

マークアップ

```
<Button x:Name="btnLoad" Grid.Column="1" Content=" Load Pdf... "
HorizontalAlignment="Right" VerticalAlignment="Top" Margin="8"
Click="btnLoad_Click" />
```

これで、UWP スタイルのアプリケーションを作成できました。次の手順では、PDF を表示するコードをアプリケーションに追加します。

手順2:C1PdfViewer アプリケーションへのコードの追加

前の手順では、新しい UWP スタイルのプロジェクトを作成し、アプリケーションに1つの **C1PDFViewer** コントロールを追加しました。この手順では、引き続き、PDF ドキュメントをアプリケーションに追加し、PDF ファイルを表示するためのコードを C1PdfViewer コントロールに追加します。

次の手順を実行します。

1. ソリューションエクスプローラーで、プロジェクト名を右クリックし、**[追加]**→**[既存の項目]**を選択します。
2. **[既存項目の追加]**ダイアログボックスで、PDF ファイル（たとえば、サンプルに含まれる C1XapOptimizer.pdf）を見つけ、**[追加]**をクリックします。
   任意の PDF ファイルを選択できますが、その場合は、以下のコードの "C1XapOptimizer.pdf" をその PDF ファイルの名前に置き換える必要があります。
3. ソリューションエクスプローラーで PDF ファイルを選択し、プロパティウィンドウでファイルの**[ビルドアクション]**を**[埋め込まれたリソース]**に設定します。
4. ページを右クリックし、**[表示]**→**[コード]**を選択してコードビューに切り替えます。
5. コードビューで、次の import 文をページの先頭に追加します。

▶ Visual Basic でコードを書く場合

Visual Basic

```
Imports C1.Xaml.PdfViewer
```

▶ C# でコードを書く場合

C#

```
using C1.Xaml.PdfViewer;
```

6. 次のコードを Page コンストラクタに追加します。

▶ Visual Basic でコードを書く場合

Visual Basic

```
Public Sub New()
Me.InitializeComponent()
Dim asm As Assembly = GetType(MainPage).GetTypeInfo().Assembly
Dim stream As Stream
stream = asm.GetManifestResourceStream("PdfViewerSamples.C1XapOptimizer.pdf")
pdfViewer.LoadDocument(stream)
End Sub
```

▶ C# でコードを書く場合

C#

```
public MainPage()
{
this.InitializeComponent();
Assembly asm = typeof(MainPage).GetTypeInfo().Assembly;
Stream stream =
asm.GetManifestResourceStream("PdfViewerSamples.C1XapOptimizer.pdf");
pdfViewer.LoadDocument(stream);
}
```

"PdfViewerSamples" をプロジェクトの名前空間の名前に置き換える必要があります。

7. 次の btnLoad_Click イベントハンドラをプロジェクトに追加します。

▶ Visual Basic でコードを書く場合

Visual Basic

```
Private Async Sub btnLoad_Click(sender As Object, e As
Windows.UI.Xaml.RoutedEventArgs)
Dim openPicker As New FileOpenPicker()
openPicker.FileTypeFilter.Add(".pdf")
Dim file As StorageFile = Await openPicker.PickSingleFileAsync()
If file IsNot Nothing Then
Dim stream As System.IO.Stream = Await file.OpenStreamForReadAsync()
pdfViewer.LoadDocument(stream)
End If
End Sub
```

▶ C# でコードを書く場合

C#

```
private async void btnLoad_Click(object sender,
Windows.UI.Xaml.RoutedEventArgse)
{
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.FileTypeFilter.Add(".pdf");
StorageFile file = await openPicker.PickSingleFileAsync();
if (file != null)
{
Stream stream = await file.OpenStreamForReadAsync();
pdfViewer.LoadDocument(stream);
}
```

```
}
```

この手順では、アプリケーションにコードを追加しました。次の手順では、アプリケーションを実行し、実行時の操作を確認します。

手順3:C1PdfViewer アプリケーションの実行

これまでに UWP スタイルのアプリケーションを作成し、外観と動作をカスタマイズしたので、次にアプリケーションを実行します。アプリケーションを実行し、**PdfViewer for UWP** の実行時の動作を確認するには、次の手順に従います。

1. **[デバッグ]**メニューから**[デバッグ開始]**を選択し、実行時にアプリケーションがどのように表示されるかを確認します。PDF の幅がビューアに合わせて調節されて PDF ファイルが表示され、アプリケーションの左上隅にページ番号が表示されることを確認します。
2. スクロールバーをクリックしてドキュメントをスクロールし、PDF ファイルを次のページにスクロールできることを確認します。
3. [**PDF のロード**]ボタンをクリックし、別の PDF ファイルを見つけて選択し、[**開く**]をクリックします。選択したファイルが C1PdfViewer コントロールにロードされることを確認します。

おめでとうございます。

これで **PdfViewer for UWP** クイックスタートは完了です。**PdfViewer for UWP** アプリケーションを作成し、C1PdfViewer コントロールをカスタマイズし、アプリケーションの実行時機能をいくつか確認することができました。

C1PdfViewer の使い方

ドキュメントの読み込み

既存の PDF ファイルを開くには、ファイルにストリームを渡して **LoadDocument** メソッドまたは **LoadDocumentAsync** メソッドを使用します。ユーザーが選択したファイルを開くには、次のコードを実行します。

C#

```
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.FileTypeFilter.Add(".pdf");
StorageFile file = await openPicker.PickSingleFileAsync();
if (file != null)
{
   Stream stream = await file.OpenStreamForReadAsync();
   pdfViewer.LoadDocument(stream);
}
```

非同期読み込み

パフォーマンスを向上させるために、**C1PdfViewer** コントロールがバックグラウンドで非同期にドキュメントを読み込むようにすることができます。.NET の await キーワードを使用すると、非同期メソッドを簡単に呼び出すことができます。ユーザーが選択したファイルを非同期に開くには、次のコードを実行します。

> **メモ:** "await" キーワードを使用するには、呼び出しを行うイベントまたはメソッドが "sync" キーワードを使用して非同期としてマークされている必要があります。

C#

```
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.FileTypeFilter.Add(".pdf");
StorageFile file = await openPicker.PickSingleFileAsync();
if (file != null)
{
  Stream stream = await file.OpenStreamForReadAsync();
  await pdfViewer.LoadDocumentAsync(stream);
}
```

暗号化されたファイルの読み込み

C1PdfViewer を使用して暗号化されたファイルは、暗号化したときのパスワードがあれば開くことができます。パスワード保護された PDF ドキュメントを読み込むには、パラメータにパスワードを指定して **LoadDocument** メソッドまたは **LoadDocumentAsync** メソッドを使用します。

C#

```
string password = "password";
await pdfViewer.LoadDocumentAsync(stream, password);
```

暗号化されたファイルと暗号化されていないファイルを一緒に開く完全な例については、トピック 保護されている可能性のあるファイルを開くを参照してください。

タッチ操作

PdfViewer for UWP は、タッチ環境に向けて最適化されており、いくつかのタッチ操作が用意されています。スマートフォンの画面を指でタッチアンドドラッグして別のドキュメント領域を表示したり、指の移動方向に基づいてドキュメントの表示部分が変化するなど、操作はかなり直感的です。

今後、**PdfViewer for UWP** のタッチ環境を強化するために、他の操作も追加される予定です。

ビューモード

C1PdfViewer は、複数の表示モードを備えており、ドキュメントを任意のスケールで表示できます。ユーザーは、ページがビューに収まるようにズームレベルを設定できます。境界によっては、1ページで表示するか、または複数ページで表示します。

ViewModes プロパティにより、ページの高さを指定して、ページを画面にどのように合わせるかを決定できます。**ViewMode** プロパティには、次のオプションがあります。

- **Normal** - 現在のズーム値を使用してページを表示します。
- **OnePage** - 1ページ全体がビューポート内に収まるように、ズーム値を自動的に更新します。
- **FitWidth** - 1ページの幅がビューポート内に収まるように、ズーム値を自動的に更新します。

次の Xaml コードは、コントロールに PDF の幅全体が表示されるようにコントロールの **ViewMode** を設定する方法を示しています。

▶ XAML でマークアップを書く場合

マークアップ

```
<PdfViewer:C1PdfViewer x:Name="pdfViewer" ViewMode="FitWidth" Grid.Row="1"
    Grid.ColumnSpan="2"/>
```

FitWidth オプションが使用されていて、Orientation が Horizontal に設定されている場合、2ページ以上表示できるか1ページしか表示できないかは、境界によって決まります。

方向

ページは、**Orientation** プロパティを使用して、PDFViewer に水平方向または垂直方向に表示できます。

タスク別ヘルプ

タスクベースのヘルプは、ユーザーの皆様が Visual Studio でのプログラミングに精通しており、C1Chart コントロールの一般的な使用方法を理解していることを前提としています。**PdfViewer for UWP** 製品を初めて使用される場合は、まず「**PdfViewer for UWP クイックスタート**」を参照してください。

このセクションの各トピックは、**PdfViewer for UWP** 製品を使用して特定のタスクを実行するための方法を提供します。

また、タスクベースのヘルプトピックは、新しいプロジェクトが既に作成されていることを前提としています。

アプリケーションリソースからドキュメントを読み込む

既存の PDF ファイルをアプリケーションと共にパッケージ化して、実行時に **C1PdfViewer** に簡単に読み込むことができます。たとえば、次の手順に従います。

1. ソリューションエクスプローラーに移動し、プロジェクト名を右クリックし、**[追加]**→**[新規フォルダ]**を選択します。新しいフォルダ名を "Resources" と指定します。
2. ソリューションエクスプローラーで**[Resources]**フォルダを右クリックし、**[追加]**→**[既存の項目]**を選択します。
3. **[既存の項目の追加]**ダイアログボックスで、PDF ファイルを見つけます。必要に応じて、ファイルの種類ドロップダウンボックスで**[すべてのファイル]**を選択して、PDF ファイルを表示します。別の PDF ファイルを選択して使用してもかまいません。
4. ソリューションエクスプローラーで、アプリケーションに追加した PDF ファイルをクリックします。この例では、このファイルの名前を **MyPdf.pdf** とします。[プロパティ]ウィンドウで、**ビルド アクション** プロパティを **Content** に設定し、**[出力ディレクトリにコピー]**項目が**[コピーしない]**に設定されていることを確認します。
5. デザインビューでプレビューをダブルクリックして、コードビューに切り替えます。
6. 次の imports 文をページの先頭に追加します。

▸ VB でマークアップを書く場合

Visual Basic
```
Imports C1.Xaml.PdfViewer
```

▸ C#でマークアップを書く場合

C#
```
using C1.Xaml.PdfViewer;
```

7. メインクラスに次のコードを追加します。

▸ VB でマークアップを書く場合

Visual Basic
```
Dim resource As StorageFile = Await
StorageFile.GetFileFromApplicationUriAsync(New Uri("ms-
appx:///Resources/MyPdf.pdf"))
Dim stream As Stream = Await resource.OpenStreamForReadAsync()
Await PdfDocument.LoadFromFileAsync(resource)
C1PdfViewer1.LoadDocument(stream)
```

▸ C#でマークアップを書く場合

C#
```
StorageFile resource = await StorageFile.GetFileFromApplicationUriAsync(new
```

```
Uri("ms-appx:///Resources/MyPdf.pdf"));
Stream stream = await resource.OpenStreamForReadAsync();
await PdfDocument.LoadFromFileAsync(resource);
C1PdfViewer1.LoadDocument(stream);
```

このコードは、**LoadDocument**メソッドを呼び出して、アプリケーションリソースストリームを渡します。

ここまでの成果

この例では、アプリケーションリソースから PDF ファイルを **C1PdfViewer** に読み込みました。コードで既存の PDF ファイルをアプリケーションと共にパッケージ化し、実行時にそのファイルを **C1PdfViewer** に読み込みました。

Web からドキュメントを読み込む

Web からファイルを読み込むには、最初に HttpClient などの非同期リクエストオブジェクトを使用して、アプリケーションにファイルをダウンロードする必要があります。次に、結果のストリームを LoadDocument メソッドまたは LoadDocumentAsync メソッドに渡します。次のコード例では、HTTP リクエストを使用しています。

C#

```
private async void LoadDocument()
    {
    // Web からファイルを読み込みます
     HttpClient client = new HttpClient();
     string url = "http://cdn.componentone.com/files/win8/Win8_UXG_RTM.pdf";
     try
     {
     var stream = await client.GetStreamAsync(new Uri(url, UriKind.Absolute));
     pdfViewer.LoadDocument(stream);
     }
     catch
     {
     var dialog = new MessageDialog("There was an error attempting to download
         the document.");
     dialog.ShowAsync();
     }
    }
```

C1PdfDocument で作成された PDF ファイルを読み込む

PDF for UWP クラスライブラリを使用して、コードで PDF ドキュメントを作成し、ストリームに保存できます。次のコードスニペットは、単純なドキュメントを作成し、それを分離ストレージに保存したり、Web に配置することなく、**C1PdfViewer** に読み込む方法を示します。

▶ VB でマークアップを書く場合

VB

```
' 新しい C1PdfDocument を作成します
Dim doc As New C1PdfDocument()
' PDF にコンテンツを追加します
doc.DrawString("Hello World!", New Font("Arial", 14), Colors.Black,
doc.PageRectangle)
```

```
' PDF をメモリストリームに保存します
Dim ms As New MemoryStream()
doc.Save(ms)
' ストリームから PDF を読み込みます
ms.Seek(0, SeekOrigin.Begin) c1PdfViewer1.LoadDocument(ms)
```

▶ C# でマークアップを書く場合

C#

```
// 新しい C1PdfDocument を作成します
C1PdfDocument doc = new C1PdfDocument();
// PDF にコンテンツを追加します
doc.DrawString("Hello World!", new Font("Arial", 14), Colors.Black,
doc.PageRectangle);
// PDF をメモリストリームに保存します
MemoryStream ms = new MemoryStream();
doc.Save(ms);
// ストリームから PDF を読み込みます
ms.Seek(0, SeekOrigin.Begin); await c1PdfViewer1.LoadDocumentAsync(ms);
```

保護されている可能性のあるファイルを開く

エンドユーザーが PDF ファイルを開くことができるようにする場合、そのファイルがパスワード保護されるかどうかを予測できないことがあります。次のサンプルの方法は、このことをチェックして、それに従ってドキュメントを開く方法を示しています。

▶ VB でマークアップを書く場合

VB

```
' 既存の PDF ファイルを開きます
Private Sub _btnOpen_Click(sender As Object, e As RoutedEventArgs)
Dim dlg = New OpenFileDialog()
dlg.Filter = "Pdf files (*.pdf)|*.pdf"
If dlg.ShowDialog().Value
Then Dim ms = New System.IO.MemoryStream()
Using stream = dlg.File.OpenRead()
stream.CopyTo(ms)
End Using LoadProtectedDocument(ms, Nothing)
End If
End Sub
```

▶ C# でマークアップを書く場合

C#

```
// 既存の PDF ファイルを開きます
void _btnOpen_Click(object sender, RoutedEventArgs e)
{
var dlg = new OpenFileDialog();
dlg.Filter = "Pdf files (*.pdf)|*.pdf";
if (dlg.ShowDialog().Value)
{
var ms = new System.IO.MemoryStream();
```

```
using (var stream = dlg.File.OpenRead()
{stream.CopyTo(ms);
}
LoadProtectedDocument(ms, null);
}
}
```

保護されたファイルを読み取る場合は、LoadProtectedDocument メソッドを呼び出します。パスワードを null にしてこのメソッドを呼び出すと、保護されていないファイルが開きます。ファイルがパスワード保護（暗号化）されている場合は、例外が生成され、キャッチされます。次に、ユーザーは、実際のパスワードの入力を要求され、メソッドが自分自身を再帰的に呼び出します。

▶ VB でマークアップを書く場合

VB

```
' パスワード保護された PDF ドキュメントを読み込みます。
    Private Sub LoadProtectedDocument(stream As System.IO.MemoryStream, password As
String)
    Try
    stream.Position = 0
    _viewer.LoadDocument(stream, password)
    Catch x As Exception
   '{
    Dim msg = "このファイルは、パスワードで保護されています。" & vbCr & vbLf & "Please provide
the password and try again."
    C1.Silverlight.C1PromptBox.Show(msg, "Enter Password", Function(text, result)
 If result = MessageBoxResult.OK Then
' ユーザーが指定したパスワードを使用して、もう一度試してください
   LoadProtectedDocument(stream, text)
End If
    '}
    'else
    '{
    ' throw;
    '}
    End Function)
    End Try
    End Sub
```

▶ C# でマークアップを書く場合

C#

```
// パスワード保護された PDF ドキュメントを読み込みます。
    void LoadProtectedDocument(System.IO.MemoryStream stream, string password)
     {
     try
     {
     stream.Position = 0;
     _viewer.LoadDocument(stream, password);
     }
     catch (Exception x)
     {
     //if (x.Message.IndexOf("password") > -1)
```

```
     //{
     var msg = "このファイルは、パスワードで保護されています。\r\nPlease provide the password and
try again.";
     C1.Silverlight.C1PromptBox.Show(msg, "Enter Password", (text, result) =>
     {
     if (result == MessageBoxResult.OK)
     {
     // ユーザーが指定したパスワードを使用して、もう一度試してください
     LoadProtectedDocument(stream, text);
     }
     });
     //}
     //else
     //{
     // throw;
     //}
     }
    }
```