RadPDFViewer For Silverlight and WPF

This tutorial will introduce the RadPDFViewer control, part of the Telerik suite of XAML controls

Setting Up The Project

To begin, open Visual Studio and click on the Telerik menu option. Under *Rad Controls For Silverlight* click on *Create New Telerik Project*. Name your project, accept Silverlight 5 and in the Project Configuration Wizard dialog check *FixedDocumentViewers* (notice that the dependent references are automatically checked as well), as shown in figure 1.

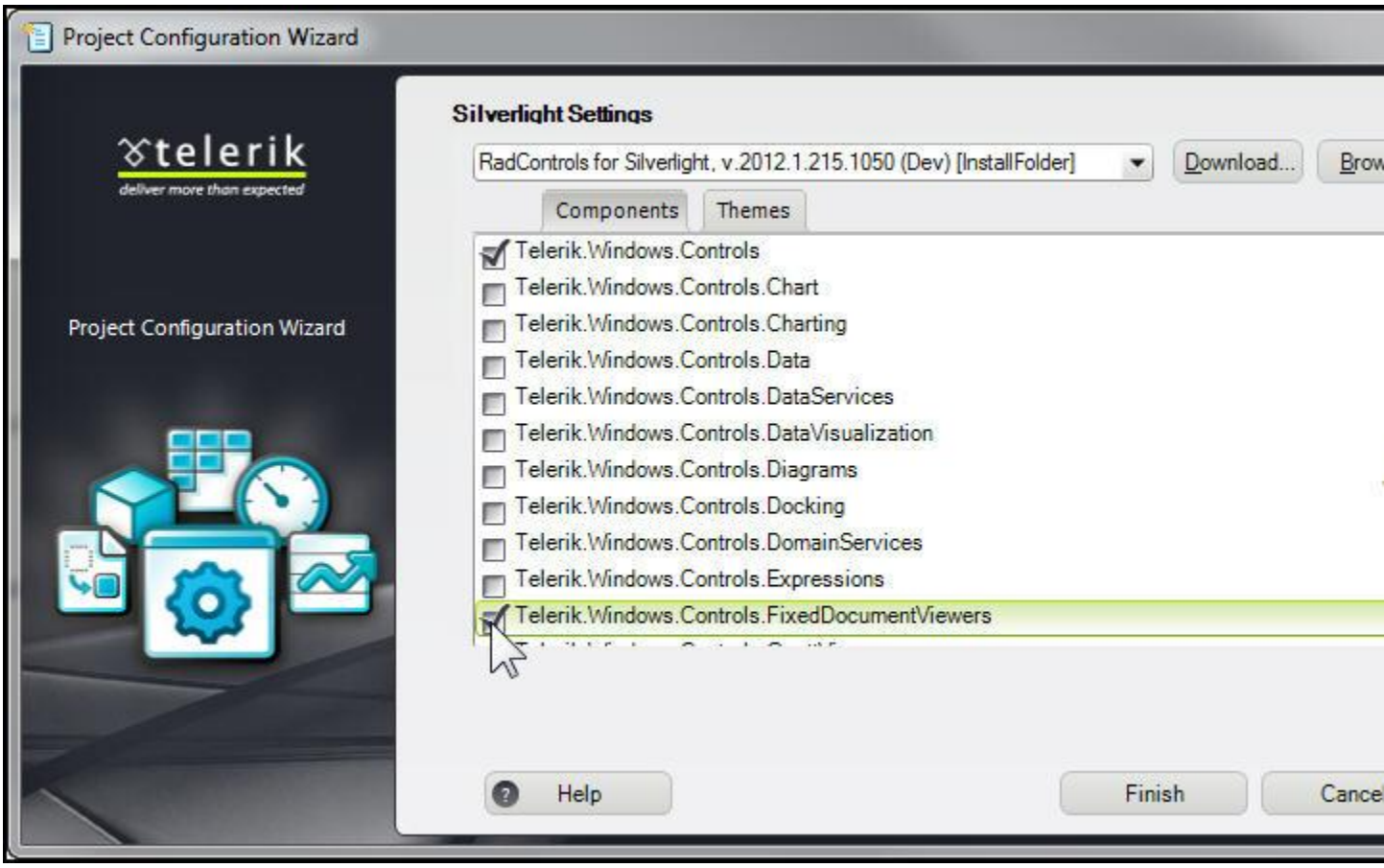

Figure 1

When you click ok, the necessary assemblies are added to the References as shown in figure 2

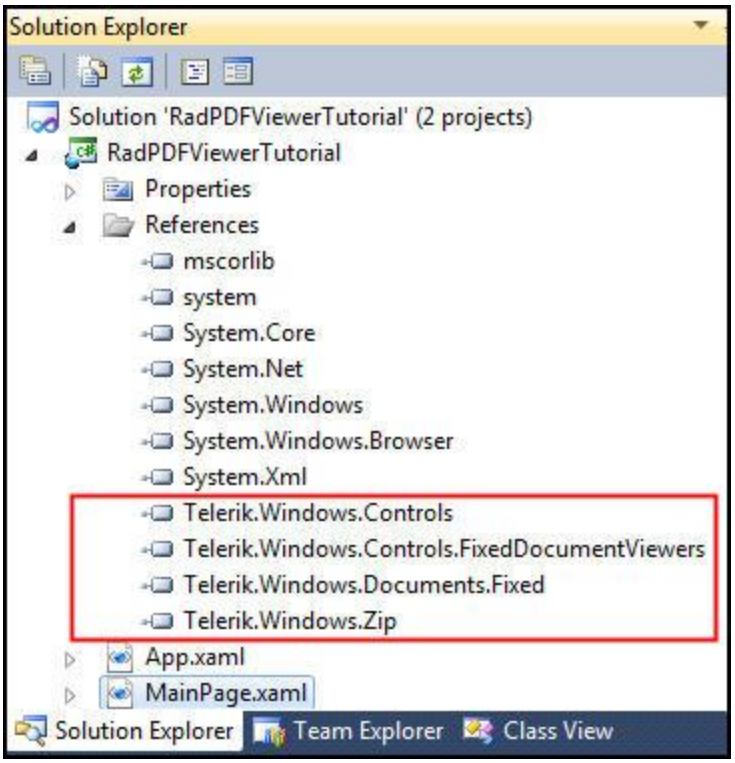

Figure 2

Your application will open to MainPage.xaml and, thanks to the Telerik Visual Studio extensions, the namespace t*elerik* will already have been created in the XAML heading.

```
<UserControl x:Class="RadPDFViewerTutorial.MainPage"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:telerik="http://schemas.telerik.com/2008/xaml/presentation"
             mc:Ignorable="d" d:DesignWidth="640" d:DesignHeight="480">
```

Create a new folder named Samples. Create a sample PDF file named Sample.pdf, or download it from here.

In either case, right click on the Samples folder and choose Add->Existing Item and add your Sample.pdf folder to the project. Set the Build Action (in the properties window) to Resource, as shown in figure 3,

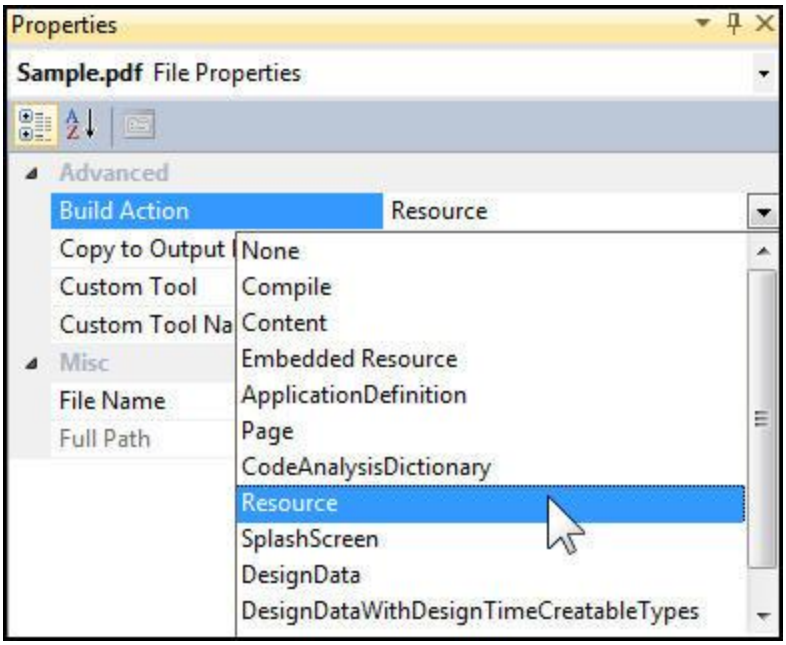

Figure 3

Add a RadPdfViewer control to your XAML file, where the key attribute is the DocumentSource,

```
<telerik:RadPdfViewer Name="xPdfViewer"
                      DocumentSource="RadPDFViewerTutorial;component/Samples/Sample.pdf"
/>
```

Run the application and the PDF file is displayed.

As an alternative to putting the DocumentSource in the XAML you can create a URI programmatically. To see this, remove the DocumentSource attribute in the XAML and in the code behind add this code,

```
public MainPage()
{
    InitializeComponent();
    this.xPdfViewer.DocumentSource =
        new PdfDocumentSource(
            new System.Uri("RadPDFViewerTutorial;component/Samples/Sample.pdf",
System.UriKind.Relative));
}
```

Notice that the path in the URI is the same as the path we had used in the XAML.

As a final alternative, you can create an explicit stream with this code,

```
var str = App.GetResourceStream(
    new System.Uri("RadPDFViewerTutorial;component/Samples/Sample.pdf",
System.UriKind.Relative)).Stream;
this.xPdfViewer.DocumentSource = new PdfDocumentSource(str);
```

Adding a Toolbar

Continue with the project we began above.

The Toolbar will give you the ability to add commands. To do this, we'll set the DataContext for the Toolbar to be bound to an instance of the RadPDFViewer.

Return to the XAML page and add two row definitions,

```
<Grid.RowDefinitions>
    <RowDefinition Height="Auto" />
    <RowDefinition Height="*" />
</Grid.RowDefinitions>
```

Return to the configuration wizard by clicking on Telerik -> RadControls for Silverlight -> Configure Project. In the wizard click on Navigation to add Telerik.Windows.Controls.Navigation to your references.

We can now add a RadToolBar to the XAML,

```
<telerik:RadToolBar DataContext="{Binding ElementName=xPdfViewer, Path=Commands}">

</telerik:RadToolBar>

<telerik:RadPdfViewer Name="xPdfViewer" Grid.Row="1"/>
```

Notice that the ElementName in the ToolBar corresponds to the Name used in the RadPdfViewer.

Comment out the code added to MainPage.xaml.cs, and in MainPage.xaml add a button to the RadToolBar,

```
<telerik:RadToolBar DataContext="{Binding ElementName=xPdfViewer, Path=Commands}">
     <telerik:RadButton Padding="4"
                        Command="{Binding OpenPdfDocumentCommand}">
         <ToolTipService.ToolTip>
                    <TextBlock Text="Open" />
                </ToolTipService.ToolTip>
         <Image
Source="/Telerik.Windows.Controls.FixedDocumentViewers;component/Images/open.png"
                Stretch="None" />
     </telerik:RadButton>
 </telerik:RadToolBar>
```

When you run the application you'll see the open image displayed, and hovering over it will display the tooltip as shown in figure 4,

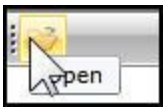

Figure 4

Clicking on the Open button will open a File-open dialog which will let you navigate to and open the Sample.pdf document in the viewer. Notice that there is *no* code-behind; just XAML.

You can add a second button to the ToolBar which will allow you to print the PDF document once it is open,

```
  <telerik:RadButton Command="{Binding PrintPdfDocumentCommand}">
      <ToolTipService.ToolTip>
          <TextBlock Text="Print" />
      </ToolTipService.ToolTip>
      <Image
Source="/Telerik.Windows.Controls.FixedDocumentViewers;component/Images/printer.png"
Stretch="None" />
  </telerik:RadButton>
</telerik:RadToolBar>
```

Because the button binds to PrintPdfDocumentCommand it invokes that command on the RadPdfViewer control, bringing up the print box as shown in figure 5,

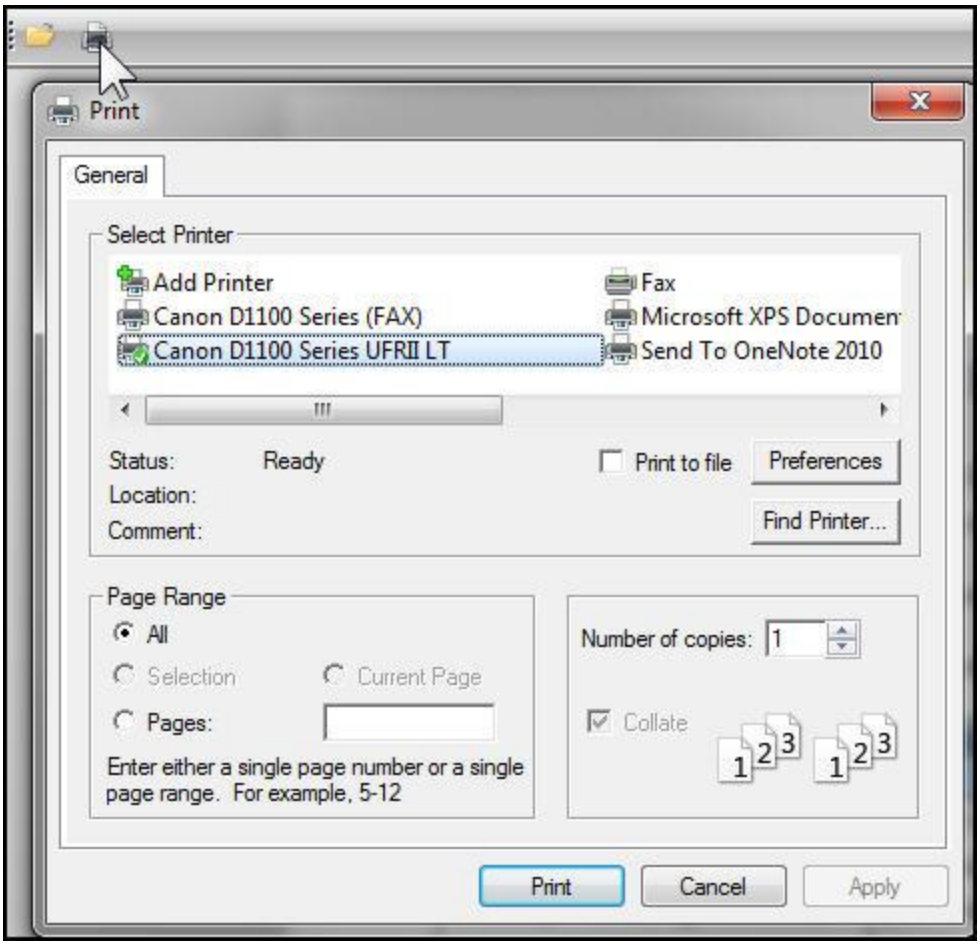

Figure 5

You can now add additional buttons and controls hooked up to various PDFViewer commands. Open the configuration wizard and add Controls.Input to support the use of a Combo box. Let's add the new controls step by step. First navigation,

```
<telerik:RadButton Command="{Binding PageUpCommand}">
    <ToolTipService.ToolTip>
        <TextBlock Text="Previous page" />
    </ToolTipService.ToolTip>
    <Image
Source="/Telerik.Windows.Controls.FixedDocumentViewers;component/Images/previous.png"
          Stretch="None" />
</telerik:RadButton>

<telerik:RadButton Command="{Binding PageDownCommand}">
    <ToolTipService.ToolTip>
        <TextBlock Text="Next page" />
    </ToolTipService.ToolTip>
    <Image
Source="/Telerik.Windows.Controls.FixedDocumentViewers;component/Images/next.png"
          Stretch="None" />
</telerik:RadButton>
```

These two buttons allow for navigation from page to page within the PDF document.

Next, we'd like a TextBox for rapid navigation across pages,

```
<TextBox Width="30"
         Margin="2"
         Text="{Binding FixedDocumentViewer.CurrentPageNumber, Mode=TwoWay}"
         HorizontalContentAlignment="Center"
         x:Name="tbCurrentPage"
         KeyDown="tbCurrentPage_KeyDown" />
<TextBlock VerticalAlignment="Center"
           Margin="2"
           Text="/" />
<TextBlock VerticalAlignment="Center"
           Margin="2"
           Text="{Binding ElementName=pdfViewer, Path=Document.Pages.Count}" />
```

The TextBlocks allow for displaying the total number of pages (e.g., 7 of 21). Notice that the TextBox has a KeyDown event, this is implemented in the code-behind,

```
private void tbCurrentPage_KeyDown(object sender, System.Windows.Input.KeyEventArgs e)
{
    TextBox textBox = sender as TextBox;
    if (textBox != null)
    {
        if (e.Key == System.Windows.Input.Key.Enter)
        {
            textBox.GetBindingExpression(TextBox.TextProperty).UpdateSource();
        }
    }
}
```

Returning to the XAML we have a separator, followed by two Buttons, one to serve for Zoom In and one to serve for Zoom Out (using the appropriate images),

```
<telerik:RadButton Command="{Binding ZoomInCommand}">
    <ToolTipService.ToolTip>
        <TextBlock Text="Zoom in" />
    </ToolTipService.ToolTip>
    <Image Source="/Telerik.Windows.Controls.FixedDocumentViewers;component/Images/zoom-
in.png"
           Stretch="None" />
</telerik:RadButton>

<telerik:RadButton x:Name="PART_btnZoomOut"
                   Command="{Binding ZoomOutCommand}">
    <ToolTipService.ToolTip>
        <TextBlock Text="Zoom out" />
    </ToolTipService.ToolTip>
    <Image Source="/Telerik.Windows.Controls.FixedDocumentViewers;component/Images/zoom-
out.png"
           Stretch="None" />
</telerik:RadButton>
```

This is followed by a Combo box which will allow for choosing the level of zoom, and will be updated when the zoom buttons are pressed,

```
<telerik:RadComboBox IsEditable="True"
                     Margin="2"
                     Width="70"
                     SelectedIndex="4"
                     IsTextSearchEnabled="True"
                     IsFilteringEnabled="True"
                     Text="{Binding FixedDocumentViewer.ScaleFactor,
                        Converter={StaticResource doubleToStringPercentConverter},
Mode=TwoWay}">
    <telerik:RadComboBoxItem Content="10%" />
    <telerik:RadComboBoxItem Content="25%" />
    <telerik:RadComboBoxItem Content="50%" />
    <telerik:RadComboBoxItem Content="75%" />
    <telerik:RadComboBoxItem Content="100%" />
    <telerik:RadComboBoxItem Content="150%" />
    <telerik:RadComboBoxItem Content="200%" />
    <telerik:RadComboBoxItem Content="500%" />
    <telerik:RadComboBoxItem Content="1000%" />
    <telerik:RadComboBoxItem Content="2000%" />
</telerik:RadComboBox>
```

Notice the data converter, this is registered just inside the definition of the RadToolBar,

```
<telerik:RadToolBar.Resources>
    <converters:DoubleToStringPercentConverter x:Key="doubleToStringPercentConverter" />
</telerik:RadToolBar.Resources>
```

And the converters namespace is registered at the very top of the file,

```
xmlns:converters="clr-
namespace:Telerik.Windows.Documents.Converters;assembly=Telerik.Windows.Controls.FixedDoc
umentViewers"
```

In short, this converter, which takes a double and returns a percentage as a string, is supplied for you.

Finally, a toggle button is used to turn panning on and off,

```
<telerik:RadToggleButton IsChecked="{Binding FixedDocumentViewer.IsInPanMode,
Mode=TwoWay}">
    <ToolTipService.ToolTip>
        <TextBlock Text="Pan" />
    </ToolTipService.ToolTip>
    <Image Source="/Telerik.Windows.Controls.FixedDocumentViewers;component/Images/hand-
free.png"
           Stretch="None" />
</telerik:RadToggleButton>
```

Integration with RadRichTextBox

Our goal is to use the RadRichTextBox to create a pdf file and then to use the PdfViewer to view the document. Let's start with a new application,

In the project configuration wizard, add RichTextBoxUI, which will add numerous other dependent components. Also add FixedDocumentViewers, for the PDF Viewer. Scroll down to the list of format providers, and add the PDF Format provider. Figure 6 shows the complete list of references added by the Telerik Wizard,

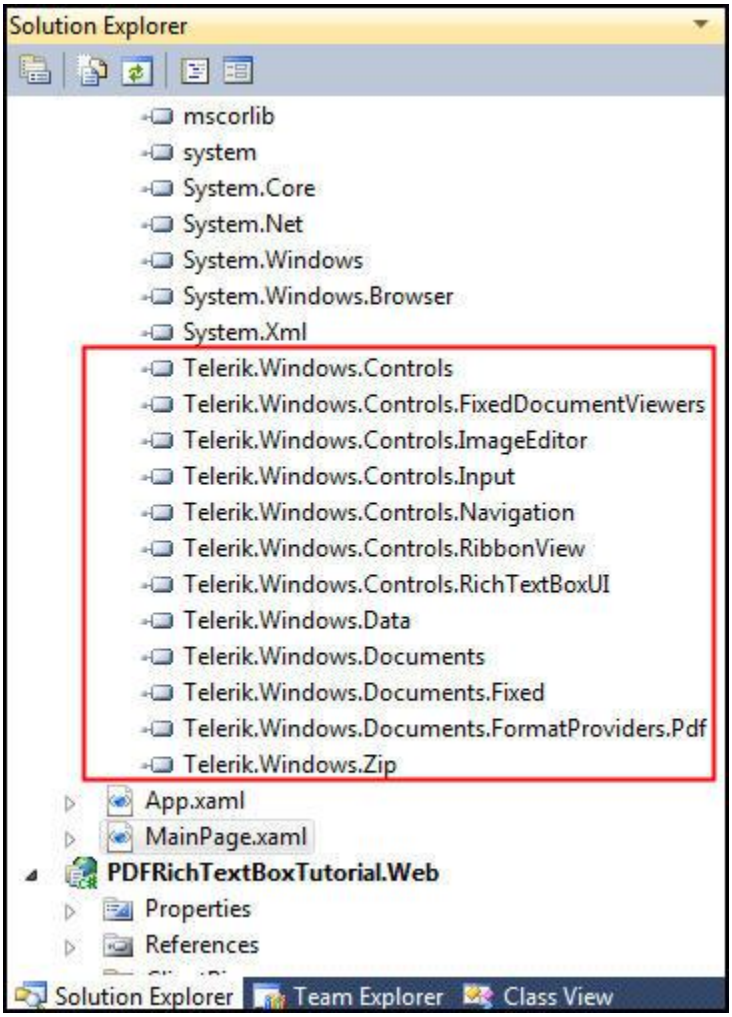

Figure 6

Create three rows in the grid,

```
<Grid.RowDefinitions>
    <RowDefinition Height="29" />
    <RowDefinition Height="*" />
    <RowDefinition Height="*" />
```

```
</Grid.RowDefinitions>
```

Add a button to export the data from the RadRichTextBox and load it in the RadPdfViewer,

```
<Button Content="Synchronize"
        Height="23"
        Name="btnSynchronize"
        Width="75"
        Click="btnSynchronize_Click" />

Add the RadRichTextBox to hold the text.

        <telerik:RadRichTextBox Name="xRichTextBox"
                        Grid.Row="1"
                        FontFamily="Arial"
                        FontSize="16"
                        DocumentInheritsDefaultStyleSettings="True"
                        IsSpellCheckingEnabled="False">
            <telerik:RadDocument LayoutMode="Paged">
                <telerik:Section PageMargin="10,10,10,10">
                    <telerik:Paragraph TextAlignment="Center">
                        <telerik:Span Text="This sample shows how you can integrate the "
/>
                        <telerik:Span FontWeight="Bold"
                              Text="RadPdfViewer" />
                        <telerik:Span Text=" and the " />
                        <telerik:Span FontWeight="Bold"
                               Text="RadRichTextBox" />
                        <telerik:Span Text=" control." />
                    </telerik:Paragraph>
                    <telerik:Paragraph>
                        <telerik:Span FontWeight="Bold"
                              Text="RadPdfViewer" />
                        <telerik:Span Text=" is a control that allows you to easily view
PDF files" />
                    </telerik:Paragraph>
                </telerik:Section>
            </telerik:RadDocument>
        </telerik:RadRichTextBox>
```

If you are unfamiliar with the RadRichTextBox you may want to check the earlier tutorials or XamlFlix videos on the topic.

Add the PdfViewer:

```
        <telerik:RadPdfViewer Name="xPDFViewer"
                      Grid.Row="2" />
```

Return to MainPage.xaml.cs to implement the export of the document of RadRichTextBox and its import in the RadPdfViewer.

First, add a MemoryStream field to keep the exported document,

```
private MemoryStream memoryStream;
```

Then, implement the synchronization logic,

```
private void btnSynchronize_Click(object sender, RoutedEventArgs e)
{
    SynchronizeDocuments();
}

private void SynchronizeDocuments()
{
    var provider = new PdfFormatProvider();
    memoryStream = new MemoryStream();
    provider.Export(this.xRichTextBox.Document, memoryStream);
    memoryStream.Flush();
    this.xPDFViewer.DocumentSource = new PdfDocumentSource(memoryStream);
    this.xPDFViewer.DocumentSource.Loaded += DocumentSource_Loaded;
}
```

Last, handle the DocumentSource_Loaded event to close the stream:

```
void DocumentSource_Loaded(object sender, EventArgs e)
{
    memoryStream.Close();
}
```

When you first run the application, you'll see we have the Synchronize button, the rich text box, and the pdf viewer, as shown in figure 7,

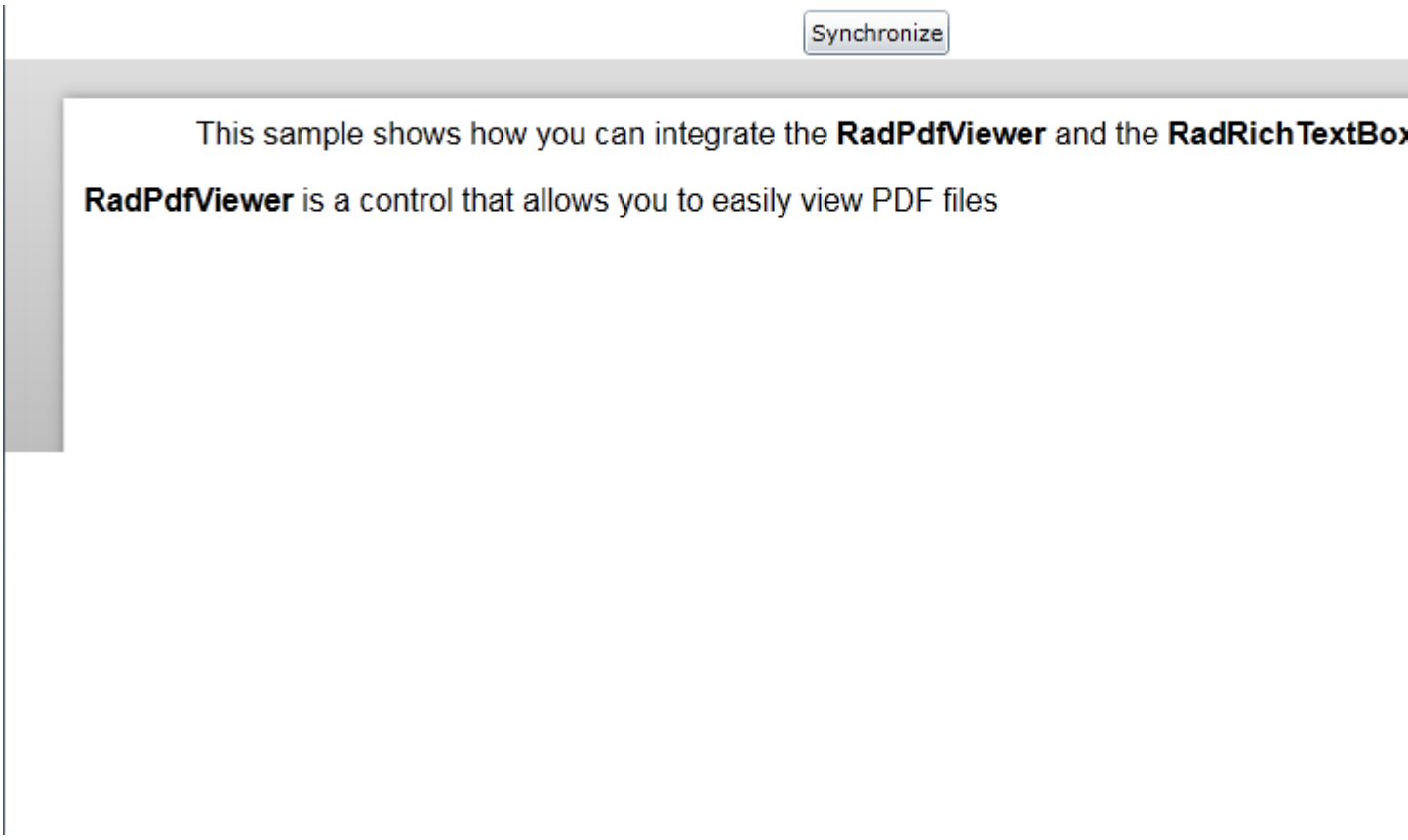

Figure 7.

You can now adjust the text, adding bold or highlighting to various words using the SelectionMiniToolBar or the FontPropertiesDialog, and then click the Synchronize button to show the document in the PdfViewer. You can see how the changes you make in the rich text box are reflected in the PDF viewer, as shown in figure 8,

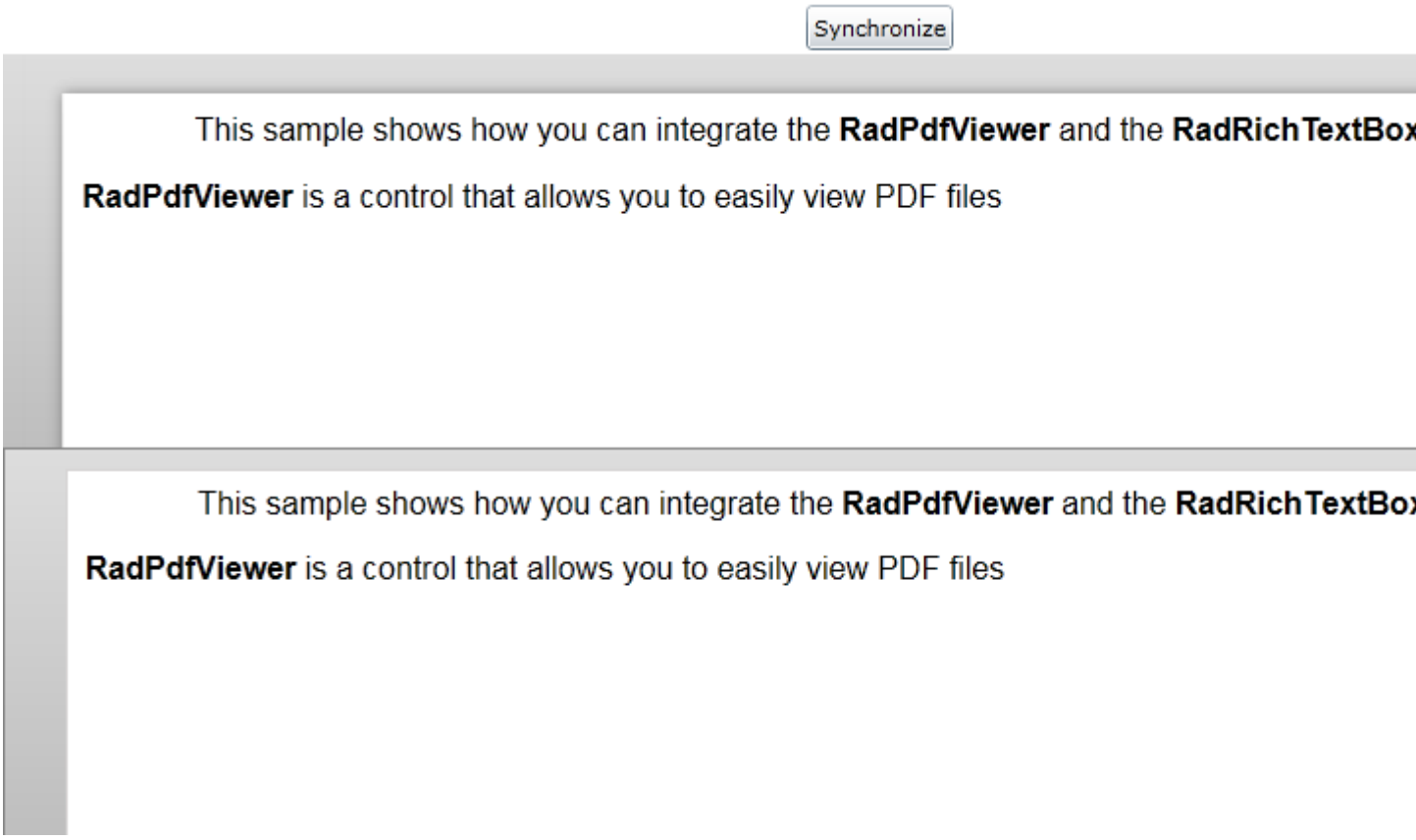
Synchronize
This sample shows how you can integrate the RadPdfViewer and the RadRichTextBox
RadPdfViewer is a control that allows you to easily view PDF files
This sample shows how you can integrate the RadPdfViewer and the RadRichTextBox
RadPdfViewer is a control that allows you to easily view PDF files

Figure 8