

[articles](#) [Q&A](#) [forums](#) [stuff](#) [lounge](#) [?](#)[Follow](#)

WPF RichText Editor

**Michael Sync,**11 Jun 2009 [CPOL](#)

Rate me! ★★★★★ 4.90 (81 votes)

Bindable WPF WYSIWYG Text Editor

[Download source - 148.29 KB](#)[Download latest version of source code \(external link\)](#)

Dependencies


[XAML to HTML Conversion Demo](#)[DelegateCommand from WPF MVVM Toolkit 0.1](#)[\(Browse Code\)](#)

Introduction

This post will give you some tips/tricks of using **RichTextbox** in WPF. As we all know, the built-in WPF **RichTextbox** doesn't provide some features that we are looking for, so if you are in need of using **RichTextbox** in a WPF project, you should know that you will need to roll your own implementation (at least) a bit. In this post, I will brief you how to make WPF **RichTextbox** bindable, how to display the HTML in WPF, how to create a **RichTextbox** Editor with toolbar.

Contents

- Bindable **RichTextbox**
- **RichText** Editor
- HTML to XAML Conversion

 wpf-rich-text-editor

Bindable Rich Textbox

A lot of people asked how to bind **RichTextbox** on the net. Yes. IMO, the Rich Textbox should be bindable but I'm not sure why **Document** property of **RichText** is not a dependency property in WPF (someone can ask me this question?) but people like

us who are using MVVM pattern need to have a binding between **RichTextbox** and the property of **ViewModel**. How are we going to make this happen?

Well, we probably need to have a custom property that can be bindable in that control so the first thing that comes to my mind is [the attached property](#). There may be a lot of definitions for it but the way I understand is that it is a custom property that can be attached to the control. For example: AA property to B Control, etc.

You can take a look at how Sam implemented the binding support for **Passwordbox** in [his post](#). (Forget about encrypting the password in memory, etc. for now.) We will follow this approach to implement the binding support in **RichTextbox** as well.

The first thing that you might notice is that **RichTextbox** has the **Document** property. So, you can create an attached property by wrapping **RichTextbox.Document** property. Please take a look at [siz's](#) implementation as below (link: [ref](#)).

Hide Copy Code

```
class RichTextboxAssistant : DependencyObject
{
    public static readonly DependencyProperty DocumentProperty =
        DependencyProperty.Register("Document",
            typeof(FlowDocument),
            typeof(RichTextboxAssistant),
            new PropertyMetadata(new PropertyChangedCallback(DocumentChanged)));

    private static void DocumentChanged(DependencyObject obj,
        DependencyPropertyChangedEventArgs e)
    {
        Debug.WriteLine("Document has changed");
    }

    public FlowDocument Document
    {
        get { return GetValue(DocumentProperty) as FlowDocument; }
        set { SetValue(DocumentProperty, value); }
    }
}
```

But..... OMG! why it's so hard to use **FlowDocument**? How come we need to call **Content.Start** and **End** just to get the text? Why not have a property called **Text** which is a **string** datatype?

Yes, it's true that using **FlowDocument** is not so simple compared to a **string** datatype. we also got the same feeling when we were implementing this feature. What did we do? We decided to change **Document** property, a **FlowDocument** type, to **"BoundDocument"** which is a **string** datatype. So, the new code will be like the one shown below. As you can see, it's a bit complicated than before since we are handling all complex things there.

Hide Shrink ▲ Copy Code

```
public static class RichTextboxAssistant
{
    public static readonly DependencyProperty BoundDocument =
        DependencyProperty.RegisterAttached("BoundDocument",
            typeof(string), typeof(RichTextboxAssistant),
            new FrameworkPropertyMetadata(null,
                FrameworkPropertyMetadataOptions.BindsTwoWayByDefault,
                OnBoundDocumentChanged));

    private static void OnBoundDocumentChanged(DependencyObject d,
        DependencyPropertyChangedEventArgs e)
    {
        RichTextBox box = d as RichTextBox;

        if (box == null)
            return;

        RemoveEventHandler(box);

        string newXAML = GetBoundDocument(d);

        box.Document.Blocks.Clear();
    }
}
```

```

if (!string.IsNullOrEmpty(newXAML))
{

using (MemoryStream xamlMemoryStream =
    new MemoryStream(Encoding.ASCII.GetBytes(newXAML)))
{

ParserContext parser = new ParserContext();
parser.XmlnsDictionary.Add
    ("", "http://schemas.microsoft.com/winfx/2006/xaml/presentation");
parser.XmlnsDictionary.Add("x", "http://schemas.microsoft.com/winfx/2006/xaml");
FlowDocument doc = new FlowDocument();

Section section = XamlReader.Load(xamlMemoryStream, parser) as Section;
box.Document.Blocks.Add(section);
}
}

AttachEventHandler(box);
}

private static void RemoveEventHandler(RichTextBox box)
{
Binding binding = BindingOperations.GetBinding(box, BoundDocument);

if (binding != null)
{
if (binding.UpdateSourceTrigger == UpdateSourceTrigger.Default ||
binding.UpdateSourceTrigger == UpdateSourceTrigger.LostFocus)
{
box.LostFocus -= HandleLostFocus;
}
else
{
box.TextChanged -= HandleTextChanged;
}
}
}

private static void AttachEventHandler(RichTextBox box)
{
Binding binding = BindingOperations.GetBinding(box, BoundDocument);
if (binding != null)
{
if (binding.UpdateSourceTrigger == UpdateSourceTrigger.Default ||
binding.UpdateSourceTrigger == UpdateSourceTrigger.LostFocus)
{
box.LostFocus += HandleLostFocus;
}
else
{
box.TextChanged += HandleTextChanged;
}
}
}

private static void HandleLostFocus(object sender, RoutedEventArgs e)
{
RichTextBox box = sender as RichTextBox;
TextRange tr = new TextRange(box.Document.ContentStart, box.Document.ContentEnd);
using (MemoryStream ms = new MemoryStream())
{
tr.Save(ms, DataFormats.Xaml);
string xamlText = ASCIIEncoding.Default.GetString(ms.ToArray());
SetBoundDocument(box, xamlText);
}
}
}

```

```

private static void HandleTextChanged(object sender, RoutedEventArgs e)
{
    // TODO: TextChanged is currently not working!
    RichTextBox box = sender as RichTextBox;
    TextRange tr = new TextRange(box.Document.ContentStart,
    box.Document.ContentEnd);

    using (MemoryStream ms = new MemoryStream())
    {
        tr.Save(ms, DataFormats.Xaml);
        string xamlText = ASCIIEncoding.Default.GetString(ms.ToArray());
        SetBoundDocument(box, xamlText);
    }
}

public static string GetBoundDocument(DependencyObject dp)
{
    return dp.GetValue(BoundDocument) as string;
}

public static void SetBoundDocument(DependencyObject dp, string value)
{
    dp.SetValue(BoundDocument, value);
}
}

```

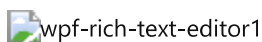
Yes. That's it. You can now simply bind this attached property with a **string** instead of a flow document.

Hide Copy Code

```
<RichTextBox attached:RichTextboxAssistant.BoundDocument="{Binding Text}" Height="92" />
```

RichText Editor

After implementing the binding support for WPF **RichTextbox**, we got a new requirement that we need to develop a RichText Editor (something like **TinyMCE**) as well. So, we quickly created a new user control called *RichTextEditor.xaml* and place a **RichTextbox** with our attached property. After a few minutes, we got a WPF **RichText** Editor as below. (As there are a lot of code snippets already in this post, I'm not going to post it here. Please feel free to take a look at *RichTextEditor.xaml* in the sample project.)



HTML to do XAML Conversion

Our manager was quite happy with our quick and cool solution for implementing WPF Rich **Textbox** so we checked-in the changes that we made to SVN, and then, the continuous integration integrated our latest changes into the new build so people from QA could start testing our new feature.

After a few hours, we started getting new bugs regarding our new **RichText** Editor from QA. Ouch!

What happened was that there is one ASP.NET website that is using the same service and same table. The ASP.NET team is using **TinyMCE**, a Javascript WYSIWYG Editor in that website so those HTML tags which are the output of that editor are being saved in the database. That's why our WPF **RichText** Editor wasn't able to render those HTML tags. The same way, their TinyMCE was also having problems with our XAML tags.



So, what should we do? Ha! I can tell what's on your mind now. Yes. a converter! What we need here is a converter that can convert HTML to XAML (vice-versa). Luckily, Microsoft provides a set of classes that can do the conversion for you. You can grab a copy of those classes from [this link](#). (Thank you! Microsoft) We embedded those classes in our application and changed our code as below to support the conversion:

```

public static string GetBoundDocument(DependencyObject dp)
{
    var html = dp.GetValue(BoundDocument) as string;
    var xaml = string.Empty;

    if (!string.IsNullOrEmpty(html))
        xaml = HtmlToXamlConverter.ConvertHtmlToXaml(html, false);

    return xaml;
}

public static void SetBoundDocument(DependencyObject dp, string value)
{
    var xaml = value;
    var html = HtmlFromXamlConverter.ConvertXamlToHtml(xaml, false);
    dp.SetValue(BoundDocument, html);
}

```

That's it. I already attached all source code in the zip file. Please feel free to download it and play as much as you like. But hey! don't forget to give the feedback if you found something uncool!.

Here is how my sample looks like. Happy WPF-ing!!! :)

 wpf-super-cool-rich-text-editor

License

This article, along with any associated source code and files, is licensed under [The Code Project Open License \(CPOL\)](#)

Share

About the Author



Michael Sync

Software Developer (Senior)
Singapore 

Follow
this Member

Michael Sync is a Microsoft MVP for Silverlight and a member of Microsoft WPF/Silverlight Insider Team.

Please find more details about me in [my blog](#).

Comments and Discussions

Add a Comment or Question



Email Alerts

Search Comments



[Image paste support](#) 

pramodgupta24 7-Oct-17 0:10

[Great Job!](#) 

Member 8241652 19-Apr-17 20:07

[Super and Sub script](#) 

Member 12201369 12-Dec-15 2:10

[Great article](#) 

dejan19dejan19 8-Jun-15 22:52

[How is it actually bound to rich text control?](#) 

shamsz 14-Apr-15 22:42

[Problem with hebrew text](#) 

galisson 26-Jan-14 19:39

[numbering/bulleted not displaying in RichTextEditor](#) 

Member 10349046 21-Oct-13 17:52

[My vote of 2](#) 

Josh_T 4-Jan-13 10:19

[System.Xaml.dll needed](#) 

superryan 4-Oct-12 8:53

Re: System.Xaml.dll needed 

Kamran Behzad 24-Jan-14 13:09

[show symbols instead of accents](#) 

Member 8044740 22-Sep-12 23:45

Re: show symbols instead of accents 

Immortal 9-Jan-13 19:42

[My vote of 5](#) 

dipal_bhavsar 11-Sep-12 18:57

[saving richtextbox content](#) 

Farhan Ghumra 19-Jul-12 19:16

[My vote of 5](#) 

Farhan Ghumra 19-Jul-12 19:15

[Could you clean unnecessary ?](#) 

Simon.P.G. 4-Mar-12 20:31

[My vote of 5](#) 

prromap 25-Jan-12 19:27

[Hypenation](#) 

brevus 19-Sep-11 21:03

[My vote of 3](#) 

urvishapandya 29-Mar-11 22:29

[My vote of 5](#) 

ChrDressler 12-Feb-11 18:32

My vote of 1 

suslicek 7-Feb-11 16:24

Doesn't seem to like horizontally oriented stack panels 

sameeraperera 22-Sep-10 14:24

My vote of 5 

koolprasadd 4-Sep-10 15:45

Downloaded Projet won't compile in VC# express [modified] 

Mr. Javaman 25-Mar-10 0:30

Re: Downloaded Projet won't compile in VC# express 

Mr. Javaman 25-Mar-10 1:15

[Refresh](#)

[1](#) [2](#) [3](#) [Next »](#)

 [General](#)  [News](#)  [Suggestion](#)  [Question](#)  [Bug](#)  [Answer](#)  [Joke](#)  [Praise](#)  [Rant](#)  [Admin](#)

Use Ctrl+Left/Right to switch messages, Ctrl+Up/Down to switch threads, Ctrl+Shift+Left/Right to switch pages.

[Permalink](#)

[Advertise](#)

[Privacy](#)

[Cookies](#)

[Terms of Use](#)

Layout: [fixed](#) | [fluid](#)

Article Copyright 2009 by Michael [Sync](#)
Everything else Copyright © [CodeProject](#),
1999-2019

Server Web01
Version 2.8.190619.2