

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

Search for articles, questions, tips

[Follow](#)

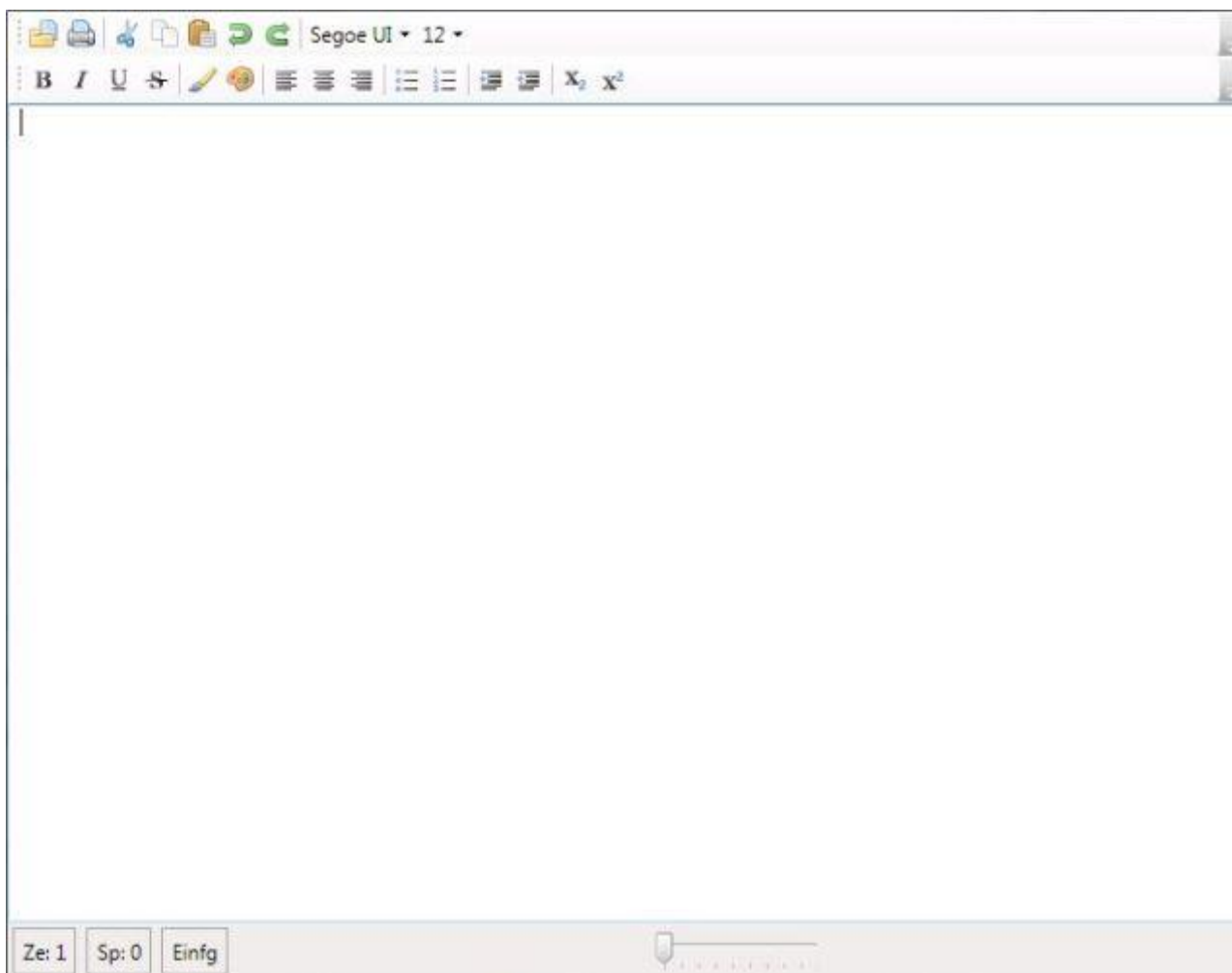
WPF RichTextEditor with Toolbar

Gregor Pross,

5 Jan 2010 [CPOL](#)

Rate me! 4.83 (20 votes)

WPF RichTextBox with standard Text-formatting possibilities

[Download source - 3.28 MB](#)[\(Browse Code\)](#)

Introduction

While playing around with WPF, I got stuck with the **RichTextBox** and their possibilities in comparison to the **RichTextBox** in WinForms.

So I tried to develop a **UserControl** with the basic text-formatting functions in WPF.

This article is it.

I want to point out that I did use the Color Picker from Sacha Barber's article:

- <http://www.codeproject.com/KB/WPF/WPFCOLORPicker.aspx>

The Icons are from:

- <http://www.famfamfam.com/lab/icons/silk>

Using the Code

So, how does it all work. It is rather simple, my RTF-Editor is a WPF UserControl-Project.

It can easily be used like I demonstrated it in my TestApplication whose XAML looks like this:

Hide Copy Code

```
<Window x:Class="TestApp.Window1"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:rtf="clr-namespace:RTFEditor;assembly=RTFEditor"
        Title="Window1" Height="300" Width="300">
    <DockPanel>
        <rtf:RTFBox></rtf:RTFBox>
    </DockPanel>
</Window>
```

The **RTFBox** class is my RTF-Editor.

The **UserInterface** consists of a **RichTextBox**, a **StatusBar** and two **ToolBar**s which hold all the buttons for text-formatting.

The first toolbars XAML looks like this:

Hide Shrink ▲ Copy Code

```
<ToolBar x:Name="ToolBarOpen" DockPanel.Dock="Top">
    <Button x:Name="ToolStripButtonOpen"
            Click="ToolStripButtonOpen_Click">
        <Image Source="Images\Open.png" Stretch="None"/>
    </Button>
    <Button x:Name="ToolStripButtonPrint"
            Click="ToolStripButtonPrint_Click">
        <Image Source="Images\Print.png" Stretch="None"/>
    </Button>
    <Separator/>
    <Button x:Name="ToolStripButtonCut"
            Command="ApplicationCommands.Cut" ToolTip="Cut">
        <Image Source="Images\Cut.png" Stretch="None"/>
    </Button>
    <Button x:Name="ToolStripButtonCopy"
            Command="ApplicationCommands.Copy" ToolTip="Copy">
        <Image Source="Images\Copy.png" Stretch="None"/>
    </Button>
    <Button x:Name="ToolStripButtonPaste">
```

```

                Command="ApplicationCommands.Paste" ToolTip="Paste">
        <Image Source="Images\Paste.png" Stretch="None"/>
</Button>
<Button x:Name="ToolStripButtonUndo"

                Command="ApplicationCommands.Undo" ToolTip="Undo">
        <Image Source="Images\Undo.png" Stretch="None"/>
</Button>
<Button x:Name="ToolStripButtonRedo"

                Command="ApplicationCommands.Redo" ToolTip="Redo">
        <Image Source="Images\Redo.png" Stretch="None"/>
</Button>
<Separator/>
<ComboBox x:Name="Fonttype" ItemsSource="{Binding Mode=OneWay,
                                                Source={StaticResource FontListKlasse}}"

                DropDownClosed="Fonttype_DropDownClosed" />
        <ComboBox x:Name="Fontheight" ItemsSource="{Binding Mode=OneWay,
                                                Source={StaticResource FontHeightKlasse}}"

                DropDownClosed="Fontheight_DropDownClosed" />

</ToolBar>

```

As you can see, it is really straightforward. I used the **ApplicationCommands** for Cut, Copy, Paste, Undo, Redo and defined the necessary Handler for Open, Print.

A bit more complex are the ComboBoxes for **FontType** and **FontHeight**. I used a OneWay Binding to static resources that I defined in my XAML like this:

Hide Copy Code

```

<UserControl.Resources>
    <ObjectDataProvider x:Key="FontListKlasse" d:IsDataSource="True"

                ObjectType="{x:Type RTFEditor:FontList}"/>
    <ObjectDataProvider x:Key="FontHeightKlasse" d:IsDataSource="True"

                ObjectType="{x:Type RTFEditor:FontHeight}"/>
</UserControl.Resources>

```

Here are the two classes **FontHeight** and **FontList** that hold the information.

Hide Shrink ▲ Copy Code

```

class FontList : ObservableCollection<string>
{
    public FontList()
    {
        foreach (FontFamily f in Fonts.SystemFontFamilies)
        {
            this.Add(f.ToString());
        }
    }
}
class FontHeight : ObservableCollection<string>
{
    public FontHeight()
    {
        this.Add("8");
        this.Add("9");
        this.Add("10");
        this.Add("11");
        this.Add("12");
        this.Add("14");
        this.Add("16");
        this.Add("18");
        this.Add("20");
        this.Add("22");
        this.Add("24");
    }
}

```

```

        this.Add("26");
        this.Add("28");
        this.Add("36");
        this.Add("48");
        this.Add("72");
    }
}

```

The second Toolbar is similar so without pasting the full XAML code:

I used ToggleButtons for the textstyle (bold, italic, ...) and textalignment(right, left, ...).

If possible, I used **EditingCommands** like this:

Hide Copy Code

```

<ToggleButton x:Name="ToolStripButtonBold"

                Command="EditingCommands.ToggleBold"

                ToolTip="Bold">
    <Image Source="Images\Bold.png" Stretch="None"/>
</ToggleButton>

```

There is no **EditingCommand** for strikethrough formatting, so I did this via **TextDecorations**:

Hide Copy Code

```

private void ToolStripButtonStrikeout_Click(object sender,
                                           System.Windows.RoutedEventArgs e)
{
    TextRange range = new TextRange(RichTextControl.Selection.Start,
                                     RichTextControl.Selection.End);

    TextDecorationCollection tdc =
        (TextDecorationCollection)RichTextControl.
            Selection.GetValue(Inline.TextDecorationsProperty);
    if (tdc == null || !tdc.Equals(TextDecorations.Strikethrough))
    {
        tdc = TextDecorations.Strikethrough;
    }
    else
    {
        tdc = new TextDecorationCollection();
    }
    range.ApplyPropertyValue(Inline.TextDecorationsProperty, tdc);
}

```

As there is no **ColorDialog** in WPF like the one in WinForms, I used one from [Sacha Barber](#) for **Textforecolor** and **Textbackgroundcolor**.

Hide Copy Code

```

private void ToolStripButtonTextcolor_Click
(object sender, RoutedEventArgs e)
{
    ColorDialog colorDialog = new ColorDialog();
    //colorDialog.Owner = this;
    if ((bool)colorDialog.ShowDialog())
    {
        TextRange range = new TextRange(RichTextControl.Selection.Start,
                                         RichTextControl.Selection.End);

        range.ApplyPropertyValue(FlowDocument.ForegroundProperty,
                                new SolidColorBrush(colorDialog.SelectedColor));
    }
}

```

For superscript and subscript exist **EditingCommands** but they do only work for **OpenType** Fonts. So instead of using them, I changed **BaselineAlignments** to display sub/superscript for all fonts like this:

```
private void ToolStripButtonSubscript_Click(object sender, RoutedEventArgs e)
{
    var currentAlignment =
        RichTextControl.Selection.GetPropertyValue
        (Inline.BaselineAlignmentProperty);

    BaselineAlignment newAlignment =
        ((BaselineAlignment)currentAlignment ==
        BaselineAlignment.Subscript) ?
        BaselineAlignment.Baseline :
        BaselineAlignment.Subscript;
    RichTextControl.Selection.ApplyPropertyValue
        (Inline.BaselineAlignmentProperty,
        newAlignment);
}
```

I added a **Statusbar** for displaying the line and column number and a zoom function. The XAML is straightforward.

To get the line number, I had to start from the beginning and count all lines to the current cursor position like this:

```
private int LineNumber()
{
    TextPointer caretLineStart =
        RichTextControl.CaretPosition.GetLineStartPosition(0);
    TextPointer p = RichTextControl.Document.ContentStart.GetLineStartPosition(0);
    int currentLineNumber = 1;

    while (true)
    {
        if (caretLineStart.CompareTo(p) < 0)
        {
            break;
        }
        int result;
        p = p.GetLineStartPosition(1, out result);
        if (result == 0)
        {
            break;
        }
        currentLineNumber++;
    }
    return currentLineNumber;
}
```

To get the column number, I could use the **GetOffsetToPosition** function like this:

```
private int ColumnNumber()
{
    TextPointer caretPos = RichTextControl.CaretPosition;
    TextPointer p = RichTextControl.CaretPosition.GetLineStartPosition(0);
    int currentColumnNumber = Math.Max(p.GetOffsetToPosition(caretPos) - 1, 0);

    return currentColumnNumber;
}
```

At this point I realized that the WinForms **RichTextBox** offers a zoom function, the WPF **RichTextBox** doesn't. I did not find a way to implement one. (Any help in this regard will be appreciated.)

After I finished with the mentioned basic **Texteditor** functions, I wanted a possibility to get the RTF-content out of the **RichTextControl**, means the text with the rich text formatting. The WPF **RichTextBox** holds no straightforward way to get this so I did this:

```

public string GetRTF()
{
    TextRange range = new TextRange(RichTextControl.Document.ContentStart,
                                     RichTextControl.Document.ContentEnd);

    // Exception abfangen für StreamReader und MemoryStream
    try
    {
        using (MemoryStream rtfMemoryStream = new MemoryStream())
        {
            using (StreamWriter rtfStreamWriter =
                    new StreamWriter(rtfMemoryStream))
            {
                range.Save(rtfMemoryStream, DataFormats.Rtf);

                rtfMemoryStream.Flush();
                rtfMemoryStream.Position = 0;
                StreamReader sr = new StreamReader(rtfMemoryStream);
                return sr.ReadToEnd();
            }
        }
    }
    catch (Exception)
    {
        throw;
    }
}

```

And the other direction looks like this:

[Hide](#) [Copy Code](#)

```

public void SetRTF(string rtf)
{
    TextRange range = new TextRange(RichTextControl.Document.ContentStart,
                                     RichTextControl.Document.ContentEnd);

    try
    {
        using (MemoryStream rtfMemoryStream = new MemoryStream())
        {
            using (StreamWriter rtfStreamWriter =
                    new StreamWriter(rtfMemoryStream))
            {
                rtfStreamWriter.Write(rtf);
                rtfStreamWriter.Flush();
                rtfMemoryStream.Seek(0, SeekOrigin.Begin);

                range.Load(rtfMemoryStream, DataFormats.Rtf);
            }
        }
    }
    catch (Exception)
    {
        throw;
    }
}

```

Your Opinion

I'm very interested in your opinion about the article and the code. Please leave some comments and let me know if the article could help you in any way.

History

- 5th January, 2010: Initial post

License

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

Share

About the Author



GregorPross

Germany

Follow
this Member

No Biography provided

Comments and Discussions

Add a Comment or Question



Email Alerts

Search Comments



First Prev Next

Would you like an About Box credit?

Member 10326256 25-Sep-17 19:37

My vote of 3

raiserle 17-Jul-17 22:31

Use DLL in vb .NET Project

trC_ 10-Jun-17 7:32

Set RTF content through MemoryStream

00zhang 19-Apr-16 7:49

Thanks You, Great Job

Henka.Programmer 7-Mar-16 18:40

Formating disappears in the beginning

Jesper Gissel 5-Aug-14 20:42

I have great difficulty to activate GregorPross' otherwise praised "WPF RichTextEditor with Toolbar" software

Member 9953121 19-Jun-13 10:45

Re: I have great difficulty to activate GregorPross' otherwise praised "WPF RichTextEditor with Toolbar" software

Member 9953121 21-Jun-13 21:36

Re: I have great difficulty to activate GregorPross' otherwise praised "WPF RichTextEditor with Toolbar" software

Member 9953121 22-Jun-13 10:28

Great, I'm using it

Jetteroh 13-Jun-12 21:11

Zoom functionality

Le Duc Anh 14-Feb-11 14:21

My vote of 5

TweakBird 22-Jan-11 20:18

Find / Replace Functionality(text with sub or super script also)

TweakBird 22-Jan-11 20:17

My vote of 5

Member 2878839 29-Sep-10 23:46

Bold / Italics toolbar buttons

Wjousts 27-Jul-10 4:07

Subscripted/superscripted characters lost from a saved rtf.

VishalvPatil 28-Jun-10 16:57

Re: Subscripted/superscripted characters lost from a saved rtf.

ketansnadar 30-Aug-12 20:12

Thanks

mpggkabol 5-Jun-10 15:47

Binding

Wizard_Memfis 4-Jun-10 3:22

Nifty...











ProtoBytes 5-Jan-10 21:23

Re: Nifty...

GregorPross 6-Jan-10 19:52

Refresh

1

 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 2010 by GregorPross
Everything else Copyright © [CodeProject](#),
1999-2019

Server Web01
Version 2.8.190619.2