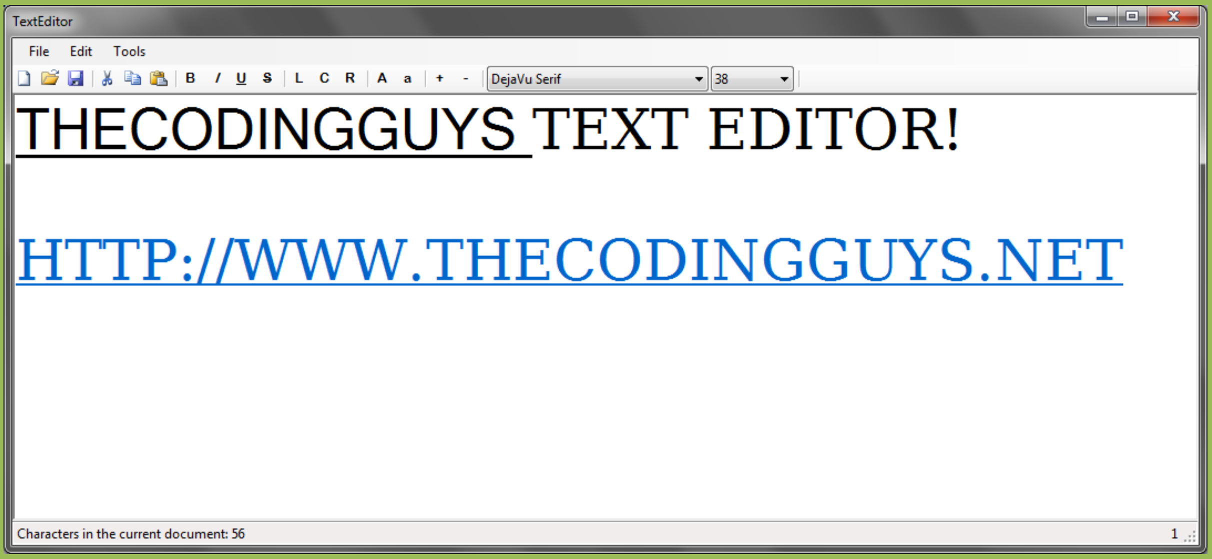


THECODINGGUYS TEXT EDITOR!

HTTP://WWW.THECODINGGUYS.NET

Characters in the current document: 56

1

C# CREATING A TEXT EDITOR

By thecodingguys

This will teach you how to make a full functional WordPad like program.

Thecodingguys.net

6/29/2012

CONTENTS

ABOUT

Published: Tuesday, 19 June 2012

Now that you have learned the basics of C# it is time to create a small project. We will create a text editor similar to windows WordPad. Since this will be a windows form application I hope you guys know how to navigate the IDE because I will not be teaching you that. I am sure you can find some useful resources on the Microsoft website.

I will split this up in two parts Part A will be design and Part B will be coding.

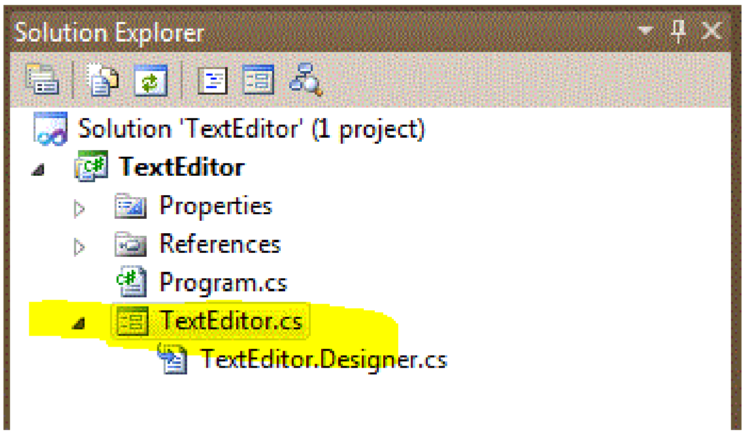
The source code and tutorial files for this can be found at http://www.thecodingguys.net/downloads

LICENSE

PART A – DESIGN

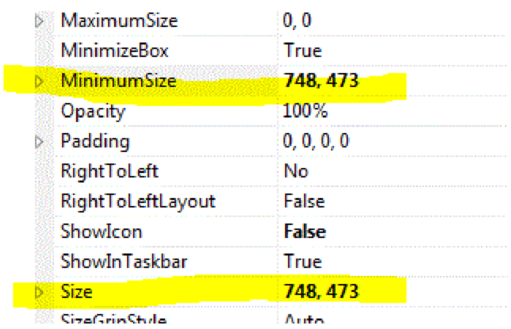
Main Form

Open up Visual Studio and create a new C# Windows Forms Application, name the application **TextEditor**. Once the project has been built on the right in the solution explorer (CTRL-R) you should see the form created with the name **Form1.cs** rename it to **Texteditor.cs**.



```
Solution Explorer
Solution 'TextEditor' (1 project)
  TextEditor
    Properties
    References
    Program.cs
    TextEditor.cs
      TextEditor.Designer.cs
```

Next resize the form so it is around **748, 473** also make sure this is applied to **Minimum Size**. (See below)

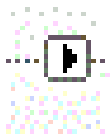


MaximumSize	0, 0
MinimizeBox	True
MinimumSize	748, 473
Opacity	100%
Padding	0, 0, 0, 0
RightToLeft	No
RightToLeftLayout	False
ShowIcon	False
ShowInTaskbar	True
Size	748, 473
SizeGripStyle	Auto

OK now it is time to start dragging controls on to the form. From the controls panel drag a **Menustrip, Toolstrip, StatusStrip** and **Richtextbox**. Now rename the Menustrip to **mainMenu** (correct casing!!), rename Toolstrip to **Tools,** rename Richtextbox to **Document** and rename statusstrip to **Status**. (Make sure you dock the Richtextbox, there is a small arrow on the upper-right hand corner)

The Main Menu

We now need to start adding things to our mainMenu, click the mainMenu and if you look towards the right you should see a small kind of triangle. As seen below.

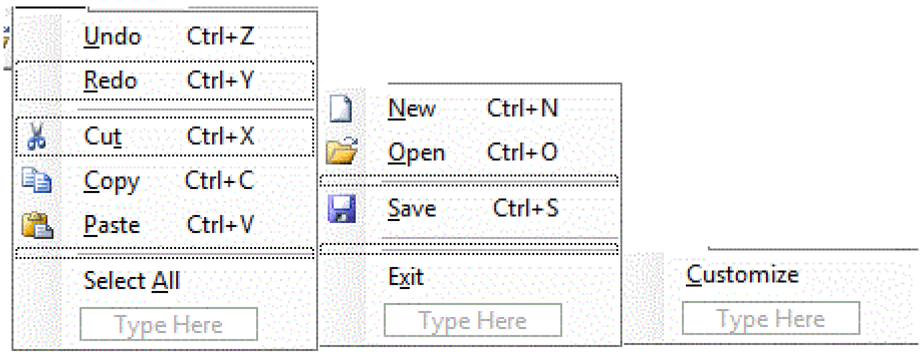


Click it then click **Insert Standard Items.** Do the same for the toolbar below. OK back to the menu since we do not need everything we need to delete a few things. By default the menu bar has the following items:

- File
- Edit
- Tools
- Help

We do not need the help menu so delete that, then under Tools delete **option** and then under file delete **Print**, **Print Preview** and **Save As**.

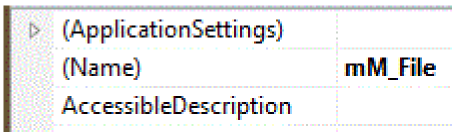
They should look like this:



(From left to right

Edit > File > Tools)

OK now it's time to rename these menu items. In the properties window rename File, Edit Tools to the following. **mM_File**, **mM_Edit** and **mM_Tools**



(ApplicationSettings)	
(Name)	**mM_File**
AccessibleDescription	

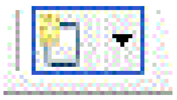
OK next we need to rename the sub menus under File, Edit and Tools. I will do one for each then you can do the rest! It's not hard.

Under **File** rename **New** to **file_New** do the same for the rest of these items. Here is the list of the named items. (Correct casing!)

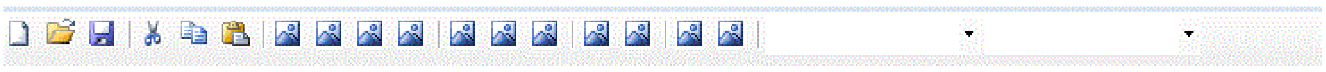
Under **File**	Under **Edit**	Under **Tools**
file_New	edit_Undo	tools_Customisation
file_Open	edit_Redo	
file_Save	edit_Cut	
file_Exit	edit_Copy	
	edit_Paste	
	edit_SelectAll	

The Tool Bar

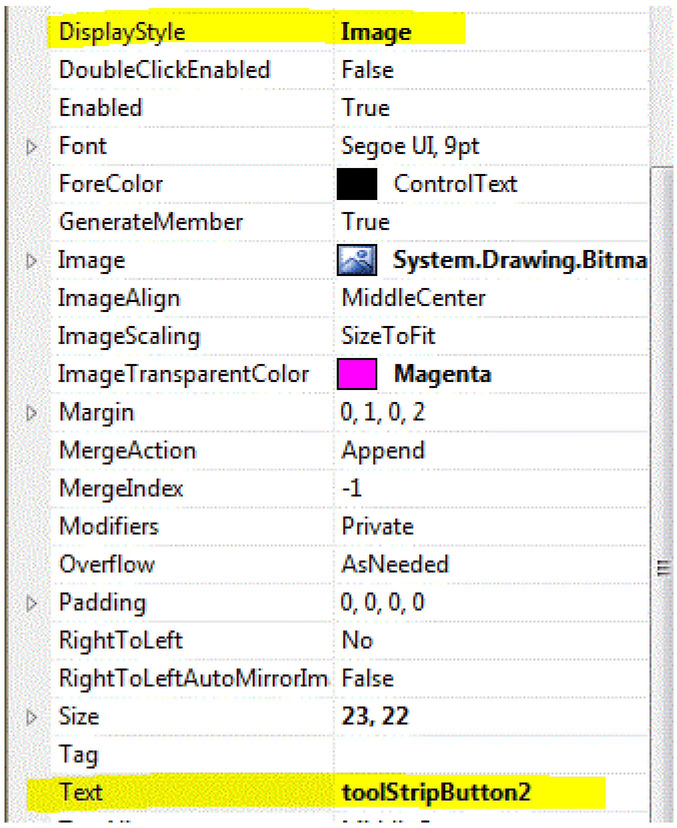
OK next we need to do the same for our toolbar make sure you have inserted the standard items. Then go ahead and add two-combo box and 11 buttons. To add controls to the toolbar click on it and you will see a little dropdown arrow like this:



Click it and add another two combo-Boxes and 11 buttons. Lay it out so it looks like this:

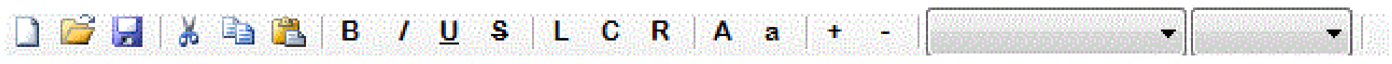


Make sure you have deleted the printer icon and help icon which is inserted by default we do not need them. Next we want to change the buttons that we just added. Click on the first button and in the properties change the **DisplayStyle** to **text** and set the **text** to **B**. (See Image)



DisplayStyle	Image
DoubleClickEnabled	False
Enabled	True
Font	Segoe UI, 9pt
ForeColor	■ ControlText
GenerateMember	True
Image	System.Drawing.Bitma
ImageAlign	MiddleCenter
ImageScaling	SizeToFit
ImageTransparentColor	■ Magenta
Margin	0, 1, 0, 2
MergeAction	Append
MergeIndex	-1
Modifiers	Private
Overflow	AsNeeded
Padding	0, 0, 0, 0
RightToLeft	No
RightToLeftAutoMirrorIm	False
Size	23, 22
Tag	
Text	toolStripButton2

Do the same for the rest, it should look like this:



OK if you did not know it's pretty obvious what were are doing here these are the buttons for Bold, Italic text and Align left, right etc. The other two are text case and zoom in/out. As you can see the buttons are bold the font for this is **Arial Rounded MT Bold** I have also styled the underline with an actual line and so on you can do this by clicking the button and then in the properties window find **Font** and change it.

Now for the combo-Box change the **DropDownStyle** to **DropDownList** and **FlatStyle** to **Standard.**

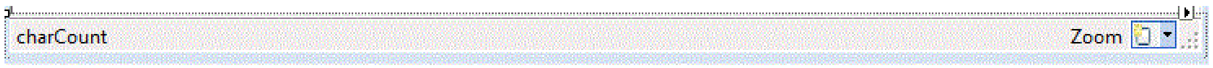
OK now it's time to rename the toolbar controls, instead of me going through the whole process here is the list of the toolbar controls with their new name. (From left to right)

tb_New	tb_AlignLeft
tb_Open	tb_AlignCenter
tb_Save	tb_AlignRight
tb_Cut	tb_ZoomIn
tb_Copy	tb_ZoomOut

tb_Paste	tb_Font (The combo-Box)
tb_Bold	tb_FontSize
tb_Italic	
tb_UnderLine	
tb_Strike	

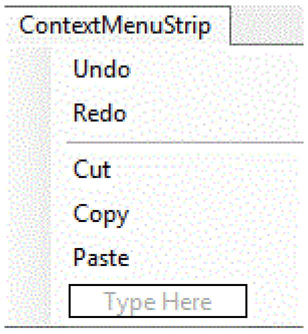
The Status Bar

OK now let's move to the status bar at the bottom. Click it and insert one label and rename it to **charCount**, then insert another label and this time in the properties scroll to the bottom and you should see something called **spring** set it to true and the new label should expand out filling up the remaining space. Next add another label and rename it to **zoom**. Now for the label which has been expanded out set its display text to nothing. The status strip should look like this:

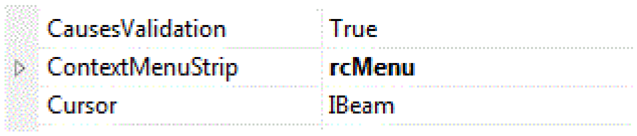


Right Click Menu

OK now go ahead and insert a **ContextMenuStrip** (it's a right click menu) this strip should dock down. Click it and add the values, Undo, Redo, Cut, Copy, and Paste.



Rename the ContextMenuStrip to **rcMenu**. Then click the Richtextbox and in the properties window there should be an option called **ContextMenuStrip** change it to **rcMenu**.

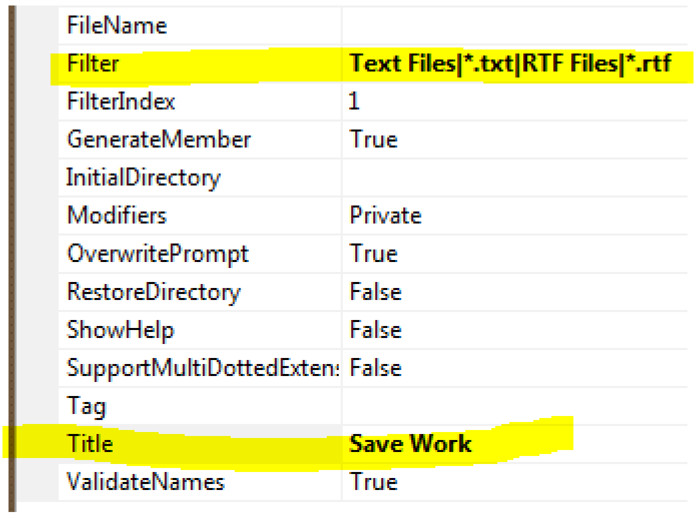


CausesValidation	True
ContextMenuStrip	**rcMenu**
Cursor	IBeam

Nearly done! Now add one timer, OpenfileDialog and SaveFileDialog. Rename timer1 to **timer** and set it to **Enabled** set the interval to **1**. Rename OpenFileDialog to **openWork** and SaveFileDialog to **saveWork**.

Save Work Dialog

Click on the Save Work Dialog that you just created and change the highlighted settings:



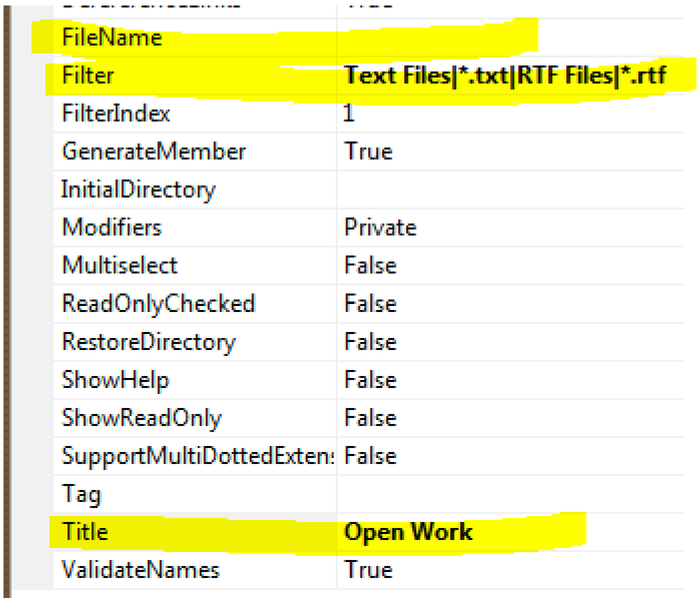
FileName	
Filter	Text Files|*.txt|RTF Files|*.rtf
FilterIndex	1
GenerateMember	True
InitialDirectory	
Modifiers	Private
OverwritePrompt	True
RestoreDirectory	False
ShowHelp	False
SupportMultiDottedExten:	False
Tag	
Title	Save Work
ValidateNames	True

OK FilterName is just the file types the user can save their work in copy this code:

Text Files|*.txt|RTF Files|*.rtf

The Title is just the title of the save work dialog, you will see this at the top when it shows.

Open Work Dialog



FileName	
Filter	Text Files|*.txt|RTF Files|*.rtf
FilterIndex	1
GenerateMember	True
InitialDirectory	
Modifiers	Private
Multiselect	False
ReadOnlyChecked	False
RestoreDirectory	False
ShowHelp	False
ShowReadOnly	False
SupportMultiDottedExten:	False
Tag	
Title	Open Work
ValidateNames	True

OK FileName is by default set to openfiledialog1, make sure this is blank. Filter is what I said above in this case it will only look for text files or rtf files. Title is also what I said above.

Phew! There we are done with the design.

PART B – CODING

OK now it's time for the fun part! We will use quite a lot of regions to organize the work. I use regions a lot because I find it keeps code neat especially if you got a lot of code. Hit **F7** and you should be in code view or go to **View > Code**. You should see this:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace TextEditor
{
    public partial class TextEditor : Form
    {
        public TextEditor()
        {
            InitializeComponent();
        }
    }
}
```

OK first thing is where it says using System.Drawing; underneath it add **using System.Drawing.Text;**

Next where it says **InitializeComponent** beneath that code block we want to make a couple of regions. Copy below:

```
#region Editor and General

#endregion


#region MainMenu

#endregion


#region Toolbar

#endregion

#region contextmenu

#endregion


#region Methods
```

```
#endregion
```

You still remember how to make regions or you forgot?

#region <region name> #endregion

OK we need more regions under **method**. Like below.

```
#region file

#endregion


#region edit

#endregion

#region tools

#endregion
```

OK then expand the region file and enter the following code:

Main Menu – File

```
     void New() {

            Document.Clear();
        }
```

File > Open. This is the code for opening a new document we use the document.load and open the file as plaintext.

```
void Open()
        {
            if (openWork.ShowDialog() == DialogResult.OK)
            {
                Document.LoadFile(openWork.FileName, RichTextBoxStreamType.PlainText);
            }
        }
```

File > Save. This is the code for saving the work just like opeing the work, we save the file as plaintext.

```
        void Save()
        {
            if (saveWork.ShowDialog() == DialogResult.OK)
            {
                try
                {

                    Document.SaveFile(saveWork.FileName,
RichTextBoxStreamType.PlainText);
                }

                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                }
            }
        }
```

File > Exit

This is the code for exiting the program.

```
        void Exit()
        {
            Application.Exit();
}
```

These methods will be called later.

Main Menu - Edit

OK under edit you want this code:

Edit > Undo

```
    void Undo()
        {
            Document.Undo();
        }
```

Edit > Redo

```
        void Redo()
        {
            Document.Redo();
        }
```

Edit > Cut

```
        void Cut()
        {
            Document.Cut();
        }
```

Edit > Copy

```
        void Copy()
        {
            Document.Copy();
        }
```

Edit >Paste

```
        void Paste()
        {
            Document.Paste();
        }
```

Edit > Select All

```
        void SelectAll()
        {
            Document.SelectAll();
        }
```

Edit > Clear All

```
        void ClearAll()
        {
            Document.Clear();
        }
```

OK this is just simple stuff we are just calling the richtextbox (document) methods. All this commands like Document.Clear, Paste etc belong to richtextbox. More information can be found on MSDN search *Richtextbox*

Main Menu - Tools

OK then under tools you want:

```
void customise()
        {
            ColorDialog myDialog = new ColorDialog();
            if (myDialog.ShowDialog() == DialogResult.OK){

                mainMenu.BackColor = myDialog.Color;
                Status.BackColor = myDialog.Color;
                Tools.BackColor = myDialog.Color;
            }

 }
```

This basically will bring up a colour dialog and change the colour of mainMenu, Status and Toolbar backcolor depending on what colour the user chose.

OK now that we have made these methods we need the mainmenu bar to function. Go to design view click on File > then double click **New.** You should see this code:

```
private void file_New_Click(object sender, EventArgs e)
        {

        }
```

All we need to do is call the method **New().** That we made earlier. It should look like this:

```
private void file_New_Click(object sender, EventArgs e)
        {
         New();
        }
```

Now when you run the program and click on File > **New** the document should clear. You need to do these for the rest of the menu items. Here is all the code:

Under File

```
  private void file_New_Click(object sender, EventArgs e)
        {
            New();
        }

        private void file_Open_Click(object sender, EventArgs e)
        {
            Open();
        }

        private void file_Save_Click(object sender, EventArgs e)
        {
            Save();
        }

        private void file_Exit_Click(object sender, EventArgs e)
        {
            Exit();
        }
```

Under Edit

```
     private void edit_Undo_Click(object sender, EventArgs e)
        {
            Undo();

        }
        private void edit_Redo_Click(object sender, EventArgs e)
        {
            Redo();
        }

        private void edit_Cut_Click(object sender, EventArgs e)
        {
            Cut();
        }

        private void edit_Copy_Click(object sender, EventArgs e)
        {
            Copy();
        }

        private void edit_Paste_Click(object sender, EventArgs e)
        {
            Paste();
        }

        private void edit_SelectAll_Click(object sender, EventArgs e)
        {
            SelectAll();
        }

        private void clearAllToolStripMenuItem_Click(object sender, EventArgs e)
        {
            ClearAll();
        }
```

Under Tools

```
private void tools_Customise_Click(object sender, EventArgs e)
        {
            customise();
        }
```

Make sure all this code goes under the region **MainMenu**.

The Context Menu Strip

OK now under the region contextmenu add the following:

```
  private void rc_Undo_Click(object sender, EventArgs e)
        {
            Undo();
        }

        private void rc_Redo_Click(object sender, EventArgs e)
        {
            Redo();
        }

        private void rc_Cut_Click(object sender, EventArgs e)
        {
            Cut();
        }

        private void rc_Copy_Click(object sender, EventArgs e)
        {
            Copy();
        }

        private void rc_Paste_Click(object sender, EventArgs e)
        {
            Paste();
        }
```

What we are doing here is just calling the methods we made before which were cut, copy, paste etc.

OK now under toolbar insert this:

The Tool Bar

OK now the first 6 icons on the tool bar are OK because we already made the code for those, we just need to call the methods copy the following, make sure this code goes in the region "ToolBar"

```
       private void tb_New_Click(object sender, EventArgs e)
        {
            New();
        }

        private void tb_Open_Click(object sender, EventArgs e)
        {
            Open();
        }

        private void tb_Save_Click(object sender, EventArgs e)
        {
            Save();
        }

        private void tb_Cut_Click(object sender, EventArgs e)
        {
            Cut();
        }

        private void tb_Copy_Click(object sender, EventArgs e)
        {
            Copy();
        }

        private void tb_Paste_Click(object sender, EventArgs e)
        {
            Paste();
        }
```

This is basically same as above. We are just calling the methods we made.

Tool Bar – Zoom In

OK this is the code for when the user clicks the plus sign (+). It will zoom in on the document.

```
        private void tb_ZoomIn_Click(object sender, EventArgs e)
        {
            if (Document.ZoomFactor == 63)
            {

                return;
            }
            else
                Document.ZoomFactor = Document.ZoomFactor + 1;
        }
```

Tool Bar – Zoom Out

OK this code will zoom out of the richtextbox when the user clicks the minus sign (-)

```
        private void tb_ZoomOut_Click(object sender, EventArgs e)
        {
            if (Document.ZoomFactor == 1)
            {

                return;


            }
            else
                Document.ZoomFactor = Document.ZoomFactor - 1;
        }
```

Tool Bar – Bold

```
        private void tb_Bold_Click(object sender, EventArgs e)
        {
            Font bfont = new Font(Document.Font, FontStyle.Bold);
            Font rfont = new Font(Document.Font, FontStyle.Regular);

            if (Document.SelectedText.Length == 0)
                return;
            if (Document.SelectionFont.Bold)
            {
                Document.SelectionFont = rfont;
            }
            else
            {
                Document.SelectionFont = bfont;
            }
        }
```

Tool Bar – Italic

```
        private void tb_Italic_Click(object sender, EventArgs e)
        {
            Font Ifont = new Font(Document.Font, FontStyle.Italic);
            Font rfont = new Font(Document.Font, FontStyle.Regular);

            if (Document.SelectedText.Length == 0)
                return;
            if (Document.SelectionFont.Italic)
            {
                Document.SelectionFont = rfont;
            }
            else
            {
                Document.SelectionFont = Ifont;
            }
        }
```

Tool Bar – Underline

```
        private void tb_UnderLine_Click(object sender, EventArgs e)
        {
            Font Ufont = new Font(Document.Font, FontStyle.Underline);
            Font rfont = new Font(Document.Font, FontStyle.Regular);

            if (Document.SelectedText.Length == 0)
                return;
            if (Document.SelectionFont.Underline)
            {
                Document.SelectionFont = rfont;
            }
            else
            {
                Document.SelectionFont = Ufont;
            }
        }
```

Tool Bar – Strikethrough

```
        private void tb_Strike_Click(object sender, EventArgs e)
        {
            Font Sfont = new Font(Document.Font, FontStyle.Strikeout);
            Font rfont = new Font(Document.Font, FontStyle.Regular);


            if (Document.SelectedText.Length == 0)
                return;
            if (Document.SelectionFont.Strikeout)
            {
                Document.SelectionFont = rfont;
            }
            else
            {
                Document.SelectionFont = Sfont;
            }
        }
```

Tool Bar – Align Left

```
  private void tb_AlignLeft_Click(object sender, EventArgs e)
        {
            Document.SelectionAlignment = HorizontalAlignment.Left;
        }
```

Tool Bar – Align Center

```
        private void tb_AlignCenter_Click(object sender, EventArgs e)
        {
            Document.SelectionAlignment = HorizontalAlignment.Center;
        }
```

Tool Bar – Align Right

```
        private void tb_AlignRight_Click(object sender, EventArgs e)
        {
            Document.SelectionAlignment = HorizontalAlignment.Right;
        }
```

Tool Bar – Upper Case

This code will transform the text to upper-case characters. We are saying “The selected document text (Document.SelectedText) transform the document selected text to upper case” (Document.SelectedText.ToUpper())

```
   private void tb_UpperCase_Click(object sender, EventArgs e)
        {
            Document.SelectedText = Document.SelectedText.ToUpper();
        }
```

Tool Bar – Upper Lower

This does the same as above but vice versa.

```

        private void tb_LowerCase_Click(object sender, EventArgs e)
        {
            Document.SelectedText = Document.SelectedText.ToLower();
        }
```

Tool Bar – Combo Boxes

OK now what we need to do is get a list of installed fonts and the numbers 10 to 75 printed into the combo-Box when the program loads.

Put this code below region methods.

```
void FontSize()
        {

            for (int fntSize = 10; fntSize <= 75; fntSize++)
            {
                tb_FontSize.Items.Add(fntSize.ToString());
            }
        }

void InstalledFonts()
        {

            InstalledFontCollection fonts = new InstalledFontCollection();

            for (int i = 0; i < fonts.Families.Length; i++)
            {
                tb_Font.Items.Add(fonts.Families[i].Name);
            }
        }
```

This is a for-iteration statement it just gets a list of installed system fonts and also gets number 10-75 printed in the combo-Boxes.

Now for this to work we need to call this method when the program loads. In the region Editor and General add the following:

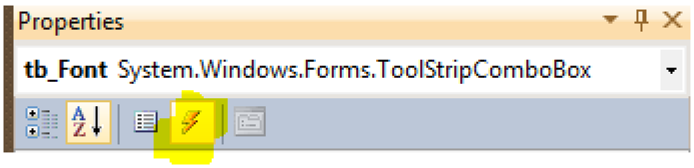
```
  private void TextEditor_Load(object sender, EventArgs e)
        {
            FontSize();
            InstalledFonts();

        }
```

This just calls the font-size and installed fonts. When you start the program in the combo boxes you should see the numbers 10-75 and the installed system fonts in the other combo box.

Tool Bar Combo Box – Installed Fonts

Now we need to make these combo-Boxes function. Go to design view, click on the first combo box in the toolbar. Go to properties and then click on events. (It has a lighting kind of icon, see image)



Now in the properties scroll down until you see **SelectedIndexChanged**. Double Clcik inside it and you should get this code.

```
  private void tb_Font_SelectedIndexChanged(object sender, EventArgs e)
        {

        }
```

Now inside it put this code:

OK so what this does that for the selected text, it changes the font, when the selection in the drop-down list changes.

```
            System.Drawing.Font ComboFonts = null;

            try
            {
                ComboFonts = Document.SelectionFont;
                Document.SelectionFont = new System.Drawing.Font(tb_Font.Text,
Document.SelectionFont.Size, Document.SelectionFont.Style);
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
```

Tool Bar Combo Box – Installed Size

OK now do the same thing for the second combo-box find it event **SelectedIndexChanged** and double click it, you should see this code:

```
        private void tb_FontSize_SelectedIndexChanged(object sender, EventArgs e)
        {

        }
```

Put this code between the code blocks.

```
            Document.SelectionFont = new Font(tb_FontSize.SelectedItem.ToString(),
int.Parse(tb_FontSize.SelectedItem.ToString()), Document.SelectionFont.Style);
```

This does the same thing as before, but this time changes the font size.

The Status Bar

Next we need to work on the status bar. The two labels we added **charCount** and **Zoom** will tell there user how many Characters in the document and how much they are zoomed into the document.

In the same region (Editor and General) add:

```
     private void Timer_Tick_1(object sender, EventArgs e)
        {
            charCount.Text = "Characters in the current document: " +
Document.TextLength.ToString();

            status_ZoomFactor.Text = Document.ZoomFactor.ToString();
        }
```

This basically means charCount (lablel).text = current characters in the document

The second just says that the label zoom factors text should be the documents zoom level Tostring. Remember ToStrings converts integer to string!

Document - Link

One last thing go to design view click on the document (richtextbox) and go to events then find the event **LinkClicked**. Double click it and then add the following bolded code:

```
private void Document_LinkClicked(object sender, LinkClickedEventArgs e)
        {
            System.Diagnostics.Process.Start(e.LinkText);
        }
```

This basically means if a link has been detected make it clickable.

You're Done

I hope you guys enjoyed this tutorial, like always feedback and criticism is welcome. I hope I have not missed anything out.

It takes a very long time to make such document please respect our copyright and spread the word about us, it really helps.

For updated and other tutorials, see:

http://www.thecodingguys.net/downloads

And

http://www.thecodingguys.net

Enjoyed it?

Follow me on:

IF YOU FOUND THIS USEFUL, PLEASE SHARE IT!