Editor Form


```
/*
Professional Windows GUI Programming Using C#
by Jay Glynn, Csaba Torok, Richard Conway, Wahid Choudhury,
Zach Greenvoss, Shripad Kulkarni, Neil Whitlow

Publisher: Peer Information
ISBN: 1861007663
*/
using System.Windows.Forms;
using System;
using System.Drawing;
using System.IO;

namespace Chapter2App
{
public class EditorForm : System.Windows.Forms.Form
{
// The RichTextBox control
private RichTextBox richTxtBox;

// constructor
public EditorForm()
{
// Instantiate the richTextBox member
this.richTxtBox = new RichTextBox();
// Set the Dock style so that the control fills the Form
this.richTxtBox.Dock = DockStyle.Fill;
// Add the RichTextBox to the Form
this.Controls.Add(this.richTxtBox);
}

public string EditText
{
```

```
get
{
return this.richTxtBox.Text;
}
set
{
richTxtBox.Text = value;
}
}

}

public class MyForm : System.Windows.Forms.Form
{
private System.Windows.Forms.MainMenu mainMenu;
private ToolBar toolBar;
private StatusBar statusBar;

// constructor
public MyForm()
{
// set the title
this.Text = “Hello world”;
// set the default size
this.Size = new System.Drawing.Size(500,500);
// set make the form non resizeable with sunken edge
//this.FormBorderStyle = FormBorderStyle.Fixed3D;

// Create the menu items
MenuItem miNew = new MenuItem(“&New”,new
EventHandler(OnMenuNew));
// adding a “-” titled menu item adds a divider on the menu
MenuItem miDash = new MenuItem(“-“);
MenuItem miOpen = new MenuItem(“&Open”,new
EventHandler(OnMenuOpen));
MenuItem miSave = new MenuItem(“&Save”,new
EventHandler(OnMenuSave));
```

```
// Create the array of menu items which will form the submenu of File
MenuItem[] fileMiArray = new MenuItem[] {miNew,miDash,miOpen,miSave};
// create the File menu Item
MenuItem miFile = new MenuItem(“&File”,fileMiArray );
// Cretat the array of menu itme for the main menu
MenuItem[] mainMiArray = new MenuItem[] {miFile};
// Create the main menu
this.mainMenu = new MainMenu(mainMiArray);

// Another way of creating MainMenu object and attaching a MenuItem
//this.mainMenu = new MainMenu();
//this.mainMenu.MenuItems.Add(miFile );

// Add the Help menu using a different route
MenuItem aboutMenuItem = new MenuItem();
aboutMenuItem.Text = “&About”;
aboutMenuItem.Click += new EventHandler(OnMenuAbout);

MenuItem helpMenuItem = new MenuItem();
helpMenuItem.Text = “&Help”;
helpMenuItem.MenuItems.Add(aboutMenuItem);

mainMenu.MenuItems.Add(helpMenuItem);

// Attach the main menu object to the form
this.Menu = this.mainMenu;

// Context Menu
//MenuItem redContext = new MenuItem(“Make Form Red”,new EventHandler(OnContextRed));
//MenuItem greyContext = new MenuItem(“Make Form Grey”,new EventHandler(OnContextGrey));

//ContextMenu colorContextMenu = new ContextMenu(new MenuItem[] {redContext,greyContext});
MenuItem colorContext = new MenuItem(“Change the color of the
```

```
Form”,new EventHandler(OnContextColor));
ContextMenu colorContextMenu = new ContextMenu(new MenuItem[]
{colorContext});

this.ContextMenu = colorContextMenu;

// Make this Form a MDI container
this.IsMdiContainer = true;

// Toolbar

// The ImageList of the toolbar
ImageList imageLst = new ImageList();
// Create Bitmap objects to add to the ImageList
//Bitmap bmpStrip = new Bitmap(“ToolBarStrip.bmp”);
Bitmap bmpStrip = new Bitmap(GetType(),”ToolBarStrip.bmp”);
imageLst.Images.AddStrip(bmpStrip);
imageLst.TransparentColor = Color.Red;

// Need the actual bmp files on the directory of the app!
//Bitmap bmpNew = new Bitmap(“NEW.bmp”);
//Bitmap bmpOpen = new Bitmap(“OPEN.bmp”);
//Bitmap bmpSave = new Bitmap(“SAVE.bmp”);
// Add the images created to the ImageList
//imageLst.Images.Add(bmpNew);
//imageLst.Images.Add(bmpOpen);
//imageLst.Images.Add(bmpSave);
// Set transparent color so that background dont show

// Create the ToolBar object
toolBar = new ToolBar();
// Toolbar belongs to this Form
toolBar.Parent = this;
// ImageList is the one we just created
toolBar.ImageList = imageLst;

/*
// Add 3 buttons
for(int i=0; i< 3;i++) { // Create the buttons ToolBarButton
```

toolBtn = new ToolBarButton(); // Show the image i from the ImageList of the toolbar toolBtn.ImageIndex = i; // Add the button to the ToolBar toolBar.Buttons.Add(toolBtn); } */ // Create the buttons ToolBarButton toolBtnNew = new ToolBarButton(); ToolBarButton toolBtnOpen = new ToolBarButton(); ToolBarButton toolBtnSave = new ToolBarButton(); // Show the image i from the ImageList of the toolbar toolBtnNew.ImageIndex = 0; toolBtnOpen.ImageIndex = 1; toolBtnSave.ImageIndex = 2; // Add the appropriate menu items toolBtnNew.Tag = miNew; toolBtnOpen.Tag = miOpen; toolBtnSave.Tag = miSave; // Add the button to the ToolBar toolBar.Buttons.Add(toolBtnNew);
toolBar.Buttons.Add(toolBtnOpen);
toolBar.Buttons.Add(toolBtnSave); // Wire the button click handler for the toolbar toolBar.ButtonClick += new ToolBarButtonClickEventHandler(ToolBarClick); statusBar = new StatusBar(); //statusBar.Text = "I am a status bar"; statusBar.ShowPanels = true; statusBar.Parent = this; // Create and the Help panel StatusBarPanel statusPanelHelp = new StatusBarPanel(); statusPanelHelp.AutoSize = StatusBarPanelAutoSize.Spring; statusPanelHelp.Text = "This is the place for help!"; statusBar.Panels.Add(statusPanelHelp); // Create and add the Date panel StatusBarPanel statusPanelDate = new StatusBarPanel(); statusPanelDate.AutoSize = StatusBarPanelAutoSize.Contents; // Get todays date DateTime dt= DateTime.Now; statusPanelDate.Text = dt.ToShortDateString(); statusBar.Panels.Add(statusPanelDate); // Wire the Closing event to the HandleClosing method // this.Closing += new System.ComponentModel.CancelEventHandler(HandleClosing); } void ToolBarClick(object sender,ToolBarButtonClickEventArgs e) { MenuItem menuItemForButton = (MenuItem) e.Button.Tag; menuItemForButton.PerformClick(); /* // Find out which button is clicked by their position switch(toolBar.Buttons.IndexOf(e.Button)) { case 0: // First button is the New button // Call the menu handler but passing null on the 2nd param this.OnMenuNew(sender,null); break; case

```
1: // Second button is the Open button this.OpenDocument(); break; case 2: // Third button is the Save button this.SaveActiveDocument(); break; } */ } void OnContextColor(object obj, EventArgs e) { ColorDialog colorDlg = new ColorDialog(); if(colorDlg.ShowDialog() == DialogResult.OK) { // Grab the third child f this Form and set its color this.Controls[2].BackColor = colorDlg.Color; // Does't work after adding the status bar since the status bar becomes the second child // Grab the second child of this Form and set its color //this.Controls[1].BackColor = colorDlg.Color; // Doesn't work with toolbar, because toolbar becomes the first child: // Grab the first child control of this Form and set its color //this.Controls[0].BackColor = colorDlg.Color; } // Doesn't work for MDI container since this Form is hidden // by a new MDI child Form managing Form //this.BackColor = Color.Red; } void OnContextRed(object obj, EventArgs e) { // Grab the first child control of this Form and set its color this.Controls[1].BackColor = Color.Red; // Doesn't work for MDI container since this Form is hidden // by a new MDI child Form managing Form //this.BackColor = Color.Red; } void OnContextGrey(object obj, EventArgs e) { // Grab the first child control of this Form and set its color this.Controls[0].BackColor = Color.Gray; // Doesn't work for MDI ... //this.BackColor = Color.Gray; } void OnMenuAbout(object obj, EventArgs e) { // Forces the context menu for the main Form to pop up at // the given position. this.ContextMenu.Show(this, new Point(20,100)); } void OnMenuNew(object obj, EventArgs e) { // Instantiate our custom form and show it EditorForm editForm = new EditorForm(); editForm.MdiParent = this; editForm.Show(); // Count the number of MDI child Forms and set the title of the // main form and the newly created Form editForm.Text = "Document " + this.MdiChildren.Length.ToString(); } void OnMenuOpen(object obj, EventArgs e) { OpenDocument(); } void OpenDocument() { OpenFileDialog openDlg = new OpenFileDialog(); openDlg.Filter = "Text Files|*.txt;*.text;*.doc|All Files|*.*"; openDlg.InitialDirectory="C:"; DialogResult dr =
```

openDlg.ShowDialog(); if(dr == DialogResult.OK) { // User click on Open // Open the file and get its content string txtFromFile = OpenFileAndReturnText(openDlg.FileName); // Instantiate our custom form and show it EditorForm editForm = new EditorForm(); editForm.MdiParent = this; // Set the title of the Form editForm.Text = openDlg.FileName; // Set the contents of the Editor Form editForm.EditText = txtFromFile; editForm.Show(); } } string OpenFileAndReturnText(string filename) { // Open the file given by param filname, read it // and return content System.IO.StreamReader reader; try { reader = new System.IO.StreamReader(filename); return reader.ReadToEnd(); } catch { MessageBox.Show("File could not be accessed"); return ""; } } void OnMenuSave(object obj, EventArgs e) { SaveActiveDocument(); } protected void SaveActiveDocument() { // Get the active MDI child Form and cast it to the custom Form EditorForm activeForm = (EditorForm) this.ActiveMdiChild; if(activeForm != null) { // Instantiate the object SaveFileDialog saveDlg = new SaveFileDialog(); // Set the suggested name saveDlg.FileName = activeForm.Text; saveDlg.InitialDirectory="C:"; // Set filter, this also sets extension saveDlg.Filter = "Text Files|*.txt; *.text; *.doc|All Files|*.*"; // Let the dialog set extension saveDlg.AddExtension = true; if(saveDlg.ShowDialog() == DialogResult.OK) { FileInfo fi = new FileInfo(saveDlg.FileName); StreamWriter sw = fi.CreateText(); sw.Write(((EditorForm)this.ActiveMdiChild).EditText); sw.Close(); } } } protected override void OnClosing(System.ComponentModel.CancelEventArgs e) { // Create an instance of the form that is to shown modally. Form modalForm = new Form(); // Create two buttons to use as the Yes and No buttons. Button yesButton = new Button (); Button noButton = new Button (); // Set the text of yes button. yesButton.Text = "Yes"; // Set the position of the button on the form. yesButton.Location = new System.Drawing.Point (10, 10); // Set the text of no button. noButton.Text = "No"; // Set the position of the button based on the location of the yes button. noButton.Location = new System.Drawing.Point

```
(yesButton.Right + 10, yesButton.Top); // Set the caption bar text of the modal form. modalForm.Text = "Are you sure you want to close??"; // set the size modalForm.Size = new System.Drawing.Size(300,100); // Add DialogResult values of the button yesButton.DialogResult = DialogResult.Yes; noButton.DialogResult = DialogResult.No; // Add the buttons to the form. modalForm.Controls.Add(yesButton); modalForm.Controls.Add(noButton); // Display the form as a modal dialog box. if(modalForm.ShowDialog()== DialogResult.No) { // setting the Cancel property of the CancelEventArgs cancels // the event and keeps the Form open e.Cancel = true; } else { // the Yes button was clicked so we need to do nothing } // Call the base method so that the event do get raised base.OnClosing(e); } static void HandleClosing(object sender, System.ComponentModel.CancelEventArgs e) { // Create an instance of the form that is to shown modally. Form modalForm = new Form(); // Create two buttons to use as the Yes and No buttons. Button yesButton = new Button (); Button noButton = new Button (); // Set the text of yes button. yesButton.Text = "Yes"; // Set the position of the button on the form. yesButton.Location = new System.Drawing.Point (10, 10); // Set the text of no button. noButton.Text = "No"; // Set the position of the button based on the location of the yes button. noButton.Location = new System.Drawing.Point (yesButton.Right + 10, yesButton.Top); // Set the caption bar text of the modal form. modalForm.Text = "Are you sure you want to close?"; // set the size modalForm.Size = new System.Drawing.Size(300,100); // Add DialogResult values of the button yesButton.DialogResult = DialogResult.Yes; noButton.DialogResult = DialogResult.No; // Add the buttons to the form. modalForm.Controls.Add(yesButton); modalForm.Controls.Add(noButton); // Display the form as a modal dialog box. if(modalForm.ShowDialog()== DialogResult.No) { // setting the Cancel property of the CancelEventArgs cancels // the event and keeps the Form open e.Cancel = true; } else { // the Yes button was clicked so we need to do nothing // the Form will close itself. } } static void Main()
```

```
{ //Instantiate the derived Form MyForm myForm = new MyForm();
//  Start  the  app  Application.Run(myForm);  }  }  }
```
Chapter2App.zip(38 k)[/csharp]