Your First VB5 App

by Chris Barlow

VB5 offers speedier results and improved features, including the Application Wizard, the Toolbox, and the Text Editor.

Sitting down to write your first application in Visual Basic has gotten easier with each new version. One of my most popular columns was the February 1996 Getting Started column ("Your First VB4 App"), which showed how to write a simple text editor in VB4 in less than five minutes. Let's build a text editor with VB5 and see what's changed. Time each stage and see how quickly you can build the complete app.

Last month, I discussed Visual Basic's new Application Wizard, one of the big enhancements in VB5. The Application Wizard helps you create a complete application with a main form, menu, toolbar, status bar, and several ancillary forms. You can use the Application Wizard to create the outline of your new app—after doing so, you simply complete some "To Do" items.

So get out your stopwatch, fire up VB5, and start the Application Wizard. Choose the Multiple Document Interface (MDI) for the user interface type. To keep this application simple, choose the defaults by clicking on the Next button until you see the "Application Wizard—Finished!" dialog caption. Then name the application TextEdit5, and click on the Finish button. This series of operations took only 20 seconds to complete on my computer.

The Application Wizard creates a Visual Basic project called TextEdit5 with two forms and a module, and the wizard displays a summary report of its actions. Press F5 to run this project, and you will see a nice-looking application with a complete menu, toolbar status bar with date and time displayed, and an MDI child window with the caption "Document 1." However, the Application Wizard isn't magic. If you type some text and click on the Save menu item in the File menu, you see a message box that says "Save Code goes here!" The Application Wizard builds the app but leaves a lot of comment lines that start with "To Do," where you will need to add code specific to your application.

Scan these To Do comments by right-clicking on frmMain in the Project window and clicking on View Code. Press Ctrl-F and type "To Do." As you click on Find Next, you'll be able to see all the To Do items—my project had 25 of them.

Before you get to work on these To Do items, take a closer look at the two forms in your project. The frmMain form has a Toolbar control and an associated ImageList control, a StatusBar control, and a CommonDialog control. The frmDocument form has only a Textbox control. You don't need to make any changes to the control on frmMain, but I suggest you use the RichTextbox control on frmDocument, rather than the Textbox control because the RichTextbox control has some useful built-in features.

Start your stopwatch again, and right-click on the Toolbox window to display the Components dialog. Find "Microsoft Rich Textbox Control 5.0" and check it to add it to your project. Then click on the Textbox control on frmDocument, delete it, and add a RichTextbox control in its place. Press the F4 key to display the Properties window, change the Name property to txtText, and set the ScrollBars property to rtfBoth. Elapsed time: one minute.

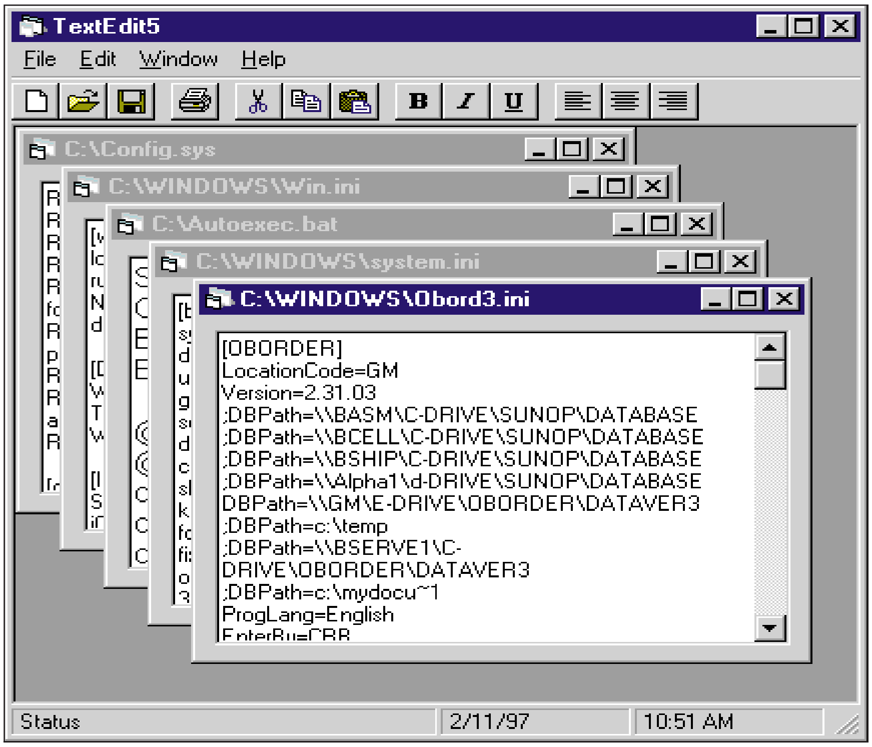
NOW WRITE SOME CODE

Remember that the frmDocument you see in your project is an MDI child form that you can use to create additional child forms when your user clicks on the New menu item on the File menu. The Application Wizard has already added this functionality in the LoadNewDoc procedure. Almost all of your project's code resides in frmMain, where the menu and toolbar are located. Display this form by right-clicking on frmMain in the Project window and clicking on View Object.

Now click on the File menu's Open menu item:

```
Private Sub mnuFileOpen_Click()
Dim sFile As String
```

Chris Barlow is president and CEO of SunOpTech, a developer of manufacturing decision-support applications, including the ObjectBank and the ObjectJob Systems. Chris and Ken Henderson hold U.S. Patent #5,550,976 related to software for decentralized distributed asynchronous object-oriented systems. Chris holds degrees from Harvard Business School and Dartmouth College, where he worked with Drs. Kemeny and Kurtz on the Basic language. Reach Chris on the Internet at ChrisB@SunOpTech.com or through SunOpTech's World Wide Web server at www.SunOpTech.com.



```
[OBORDER]
LocationCode=GM
Version=2.31.03
;DBPath=\\BASM\C-DRIVE\SUNOP\DATABASE
;DBPath=\\BCELL\C-DRIVE\SUNOP\DATABASE
;DBPath=\\BSHIP\C-DRIVE\SUNOP\DATABASE
;DBPath=\\Alpha1\d-DRIVE\SUNOP\DATABASE
DBPath=\\GM\E-DRIVE\OBORDER\DATAVER3
;DBPath=c:\temp
;DBPath=\\BSERVE1\C-
DRIVE\OBORDER\DATAVER3
;DBPath=c:\mydocu~1
ProgLang=English
EnterBu=CBB
```

FIGURE 1 ***The Three-and-a-Half-Minute Text Editor.*** *In a very short time, you can create a useful tool and wow your non-Visual Basic programmer friends.*

```
With dlgCommonDialog
   'To Do
   'set the flags and attributes of
   'the common dialog control
   .Filter = "All Files (*.*)|*.*"
   .ShowOpen
   If Len(.filename) = 0 Then
      Exit Sub
   End If
   sFile = .filename
End With
'To Do
'process the opened file
End Sub
```

Replace the last two lines in the procedure with this code. The code calls the LoadFile method of the RichTextbox control of the active document form:

```
ActiveForm.txtText.LoadFile (sFile)
ActiveForm.caption = sFile
```

The form's caption property is a convenient place to store the path and file name being edited. You can use this property when you need to save the file.

The File menu has three different "Save" menu items: "Save" to write the document back with the name on the caption, "Save As" to give the file a new name, and "Save All" to save all open documents. If the caption starts with "Document," you can assume that the user has not assigned a file name yet. Add this code to the mnuFileSave_click event:

```
Dim sFile As String
If Left(ActiveForm.Caption, 8) = _
   "Document" Then
   mnuFileSaveAs_Click
Else
   sFile = ActiveForm.Caption
   ActiveForm.txtText.SaveFile (sFile)
End If
```

Put this code in the mnuFileSaveAs_click event. Notice that this code uses the ShowOpen method of the CommonDialog control to let the user choose the file name. Simply set the caption property and call the mnuFileSave_click event to actually save the file:

```
Dim sFile As String
With dlgCommonDialog
   .Filter = "All Files (*.*)|*.*"
   .ShowOpen
   If Len(.filename) = 0 Then
      Exit Sub
   End If
   sFile = .filename
End With
ActiveForm.Caption = sFile
mnuFileSave_Click
```

My stopwatch shows that it took another 90 seconds to type those 18 lines of code. Elapsed time: 1:30.

You now have a working text editor that can load and save multiple document files. I'll give you a hint on the Save All menu item—it uses the Forms collection to loop through all frmDocument forms. For details, see the enhanced Text Editor available on the Premier Level of The Development Exchange (for details, see the Code Online box at the end of this column).

TOOLBAR FORMAT CONTROL

Now you can activate the format buttons on the toolbar to format the text in the active document. The RichTextbox control has several properties that begin with "Sel…" that control the format of the selected text. For example, SelBold returns True if the selected text's font is bold. You can add code in each Case statement in the tbToolBar_ButtonClick event to toggle the appropriate format property:

```
Case "Bold"
ActiveForm.txtText.SelBold = Not ActiveForm.txtText.SelBold
Case "Italic"
ActiveForm.txtText.SelItalic = Not _
   ActiveForm.txtText.SelItalic
Case "Underline"
ActiveForm.txtText.SelUnderline = Not _
   ActiveForm.txtText.SelUnderline
```

Copy this line of code under the Italics Case statements, and change them to set the SelItalic and SelUnderline properties. The Left, Center, and Right alignment buttons are just as easy. Simply set the SelAlignment property to rtfLeft. All you have to do then is quickly copy and paste for Center and Right—remember, the stopwatch is running:

```
Case "Left"
ActiveForm.txtText.SelAlignment = rtfLeft
Case "Center"
ActiveForm.txtText.SelAlignment = rtfCenter
Case "Right"
ActiveForm.txtText.SelAlignment = rtfRight
```

You can use the SelText property of the RichTextbox control to get or set the text the user selects. You can use this property to activate the Cut, Copy, and Paste toolbar buttons using Visual Basic's Clipboard object's GetText and SetText methods. The copy function simply places the selected text in the clipboard. The cut function first calls the copy function to place the text in the clipboard, and then sets the selected text to an empty string.

Finally, the paste function uses the Clipboard object's GetText method to insert the text in the RichTextbox control at the insertion point, replacing any selected text:

```
Private Sub mnuEditCopy_Click()
Clipboard.SetText ActiveForm.txtText.SelText
End Sub

Private Sub mnuEditCut_Click()
mnuEditCopy_Click
ActiveForm.txtText.SelText = ""
End Sub

Private Sub mnuEditPaste_Click()
ActiveForm.txtText.SelText = Clipboard.GetText
End Sub
```

My stopwatch shows an elapsed time of 3:20 to get to this point—not too bad for a full-featured text editor (see Figure 1). Show that to COBOL and C programmers!

There are still a few To Do items in the code, such as the Send menu item and the most recently used menu items on the File menu. For the complete source code listing, go to the Registered Level of The Development Exchange (DevX). Take a look at the Premier Level of DevX for a more complete implementation of this text editor, which includes the Save All menu item, the Find menu item, and the Most Recently Used list. E-mail me your enhancements along with your elapsed time, and I'll include a few of the best in a future column. ☒

Code Online

You can find all the code published in this issue of VBPJ *on The Development Exchange (DevX) at http://www.windx.com. All the listings and associated files essential to the articles are available for free to Registered members of DevX, in one ZIP file. This ZIP file is also posted in the Magazine Library of the* VBPJ *Forum on CompuServe. DevX Premier Club members ($20 for six months) can get each article's listings in a separate file, as well as additional code and utilities for selected articles, plus archives of all code ever published in* VBPJ *and* Microsoft Interactive Developer *magazines.*

Your First VB5 App

Locator+ Codes

Listings ZIP file (free Registered Level): VBPJ0597

✪ *Listings for this article plus a more complete implementation of the text editor (subscriber Premier Level): GS0597P*