Discover the Application Wizard

by Chris Barlow

Visual Basic 5.0's Wizard makes it easy for you to get started.

Microsoft has done a great job of making Visual Basic 5.0 easy for the new programmer—hardly a trivial task because people come to Visual Basic from so many directions. Perhaps you're a user who first heard about Visual Basic when you used some of the thousands of applications written in VB. Then you read that Visual Basic is the easiest way to write powerful Windows applications. Perhaps you used Visual Basic for Applications in a product such as Microsoft Office to record some Visual Basic macro code.

You might be a mainframe programmer who always wanted to write your own robust, standalone Windows applications but didn't know where to start. Maybe you even tried prior versions of Visual Basic and were intimidated by the blank form displayed when you clicked on the New Project button—how do you know what to do next?

Visual Basic's new Application Wizard makes it simple to get started on your first Visual Basic application. The Application Wizard helps you create a complete application with a main form and several ancillary forms. The main form includes a toolbar, status bar, and complete application menu. The Wizard writes some of the code to activate menu and toolbar events, and it adds comments to indicate where you need to add code to activate a program feature.

The Wizard makes a great learning device. Let it build several different kinds of applications, and then single-step through each application's code. Notice the way the menus are organized, the way the toolbar button calls the menu procedures, and the way the data access forms operate. Try changing some of the code, adding menu items, and adding forms. Exploring the many features of the Wizard is one of the best ways I know to learn Visual Basic.

When you start the Application Wizard, you must choose a user interface type (see Figure 1). You can choose from three different types of user interface: Single Document Interface (SDI), Multiple Document Interface (MDI), or Explorer Style (ESI).

Choosing SDI creates an application with a single main form containing menu items, a toolbar, and a status bar. You should use SDI for an application such as Notepad, which displays a single set of user data.

Choosing MDI creates an application with a single main form and multiple child forms. Like the SDI application, the main form contains menu items, a toolbar, and a status bar. However, when you click on the File menu's New menu item, the

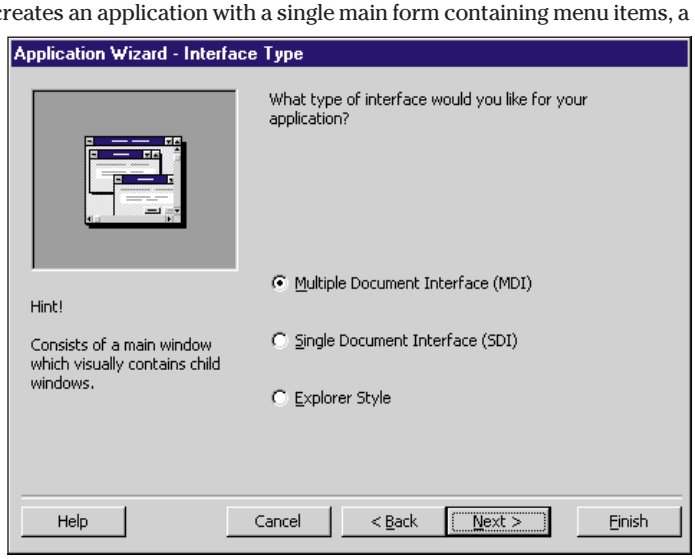

FIGURE 1 ***Pick the User Interface.*** *The Application Wizard lets you choose from three user interface types. Try them all and see how the menu and controls differ.*

Chris Barlow is president and CEO of SunOpTech, a developer of manufacturing decision-support applications, including the ObjectBank and the ObjectJob Systems. He 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.

application creates another instance of the child form. A TextBox control lies on each child form. The main form features a Window menu that allows you to view, cascade, or tile the child forms. You should use MDI for an application such as Word, which must have several different documents open at the same time.

Choosing Explorer Style creates an application with a single main form containing menu items, a toolbar, and a status bar. However, it also has a TreeView and a ListView control separated by a splitter bar, so you can write an application with an interface like the Windows Explorer.

CHOOSE YOUR MENUS

The next step in the Application Wizard lets you choose the menus you want on the form (see Figure 2). The great thing about these menus is that the Wizard adds almost 30 standard menu items to the form. For example, the View menu has menu items to control the display of the toolbar and status bar. When you check these menu items, the associated object is displayed. The Wizard adds code to these Click events to toggle the Checked property of the menu item and toggle the Visible property of the object:

```
Private Sub mnuViewToolbar_Click()
If mnuViewToolbar.Checked Then
   tbToolBar.Visible = False
   mnuViewToolbar.Checked = False
Else
   tbToolBar.Visible = True
   mnuViewToolbar.Checked = True
End If
End Sub
```

These menu items use standard nomenclature and standard shortcut keys. The File menu's Open menu item is always named mnuFileOpen and always has a Ctrl-O shortcut key combination. The mnuFileOpen_Click event already contains this code to display the File Open dialog from the CommonDialog control that the Wizard added to the form:

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

Notice the To Do comments on the fourth and 14th lines. These comments are typical of the code the Application Wizard generates. After building your project, you can search for "To Do" to find where you need to add code. Some procedures, such as mnuViewToolbar_Click, need no additional code. Other procedures, such as the File menu's Print menu item, simply display a message box reminding you "Print code goes here."

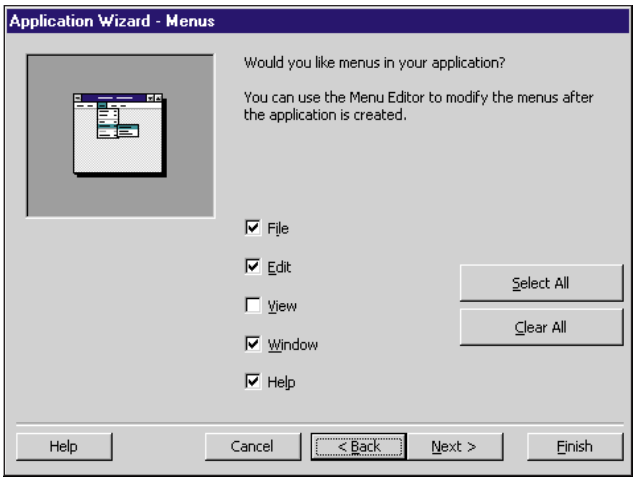

FIGURE 2 ***Choose Your Menu.*** *Menus have always been tedious to build. The Application Wizard creates a complete set of application menus, most with code to handle much of the functionality.*

VB'S NEW APPLICATION WIZARD MAKES IT SIMPLE TO GET STARTED ON YOUR FIRST APPLICATION.

The next step in the Application Wizard lets you choose if you want to store all your control captions in a Resource (RES) file. If you choose this option, you need to specify the location for the Recourse (RC) file that the Wizard compiles into the RES file for your project. Rather than setting the form caption, menu captions and control captions, and tooltips properties directly, the Wizard creates an RC file similar to this excerpt:

```
STRINGTABLE DISCARDABLE
BEGIN
      1000     "&File"
      1001     "&Open"
      1002     "&Find"
      1003     "Sen&d to"
      1004     "&New"
      1005     "&Delete"
      1006     "Rena&me"
      1007     "Propert&ies"
      1008     "&Close"
      1009     "&Edit"
      1010     "&Undo"
      1011     "Cu&t"
      1012     "&Copy"
      1013     "&Paste"
      1014     "Paste &Special..."
      1015     "Select &All"
      1016     "&Invert Selection"
```

The numeric ID numbers, rather than the text, are written into the Caption or Tag properties of the form, menu, and control objects. For example, the Open menu item on the File menu has a caption of "1001" rather than "Open" (see Figure 3). Place these strings in a string table and compile them into a resource file, rather than placing them directly onto your forms, to easily create an application that displays forms in multiple languages. You can do this by expanding the string table with additional sets of strings in each language.

In the Form_Load event, the Wizard adds code to read the strings from the resource files using the built-in LoadResString function. To load the form caption, for example, the procedure converts the form's Tag property to an integer to call the LoadResString function and set the result in the form's Caption property:

```
'set the form's caption
frm.Caption = _
   LoadResString(CInt(frm.Tag))
```

Similarly, Label and Menu controls are part of the form's Controls collection, so the procedure can loop through the Controls collection (using the For Each…Next statement) and retrieve the proper caption. Notice how you use the TypeName function to determine the type of control:

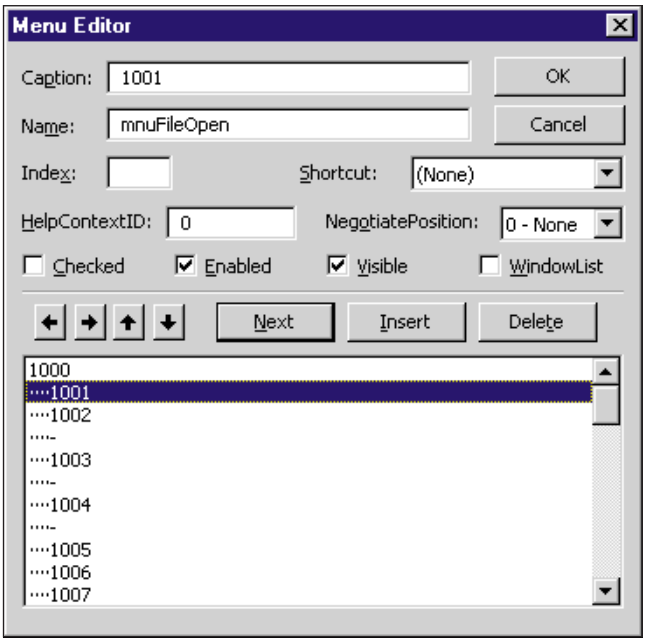

FIGURE 3 ***Resource Strings.*** *The Wizard can use a string table in a resource file to fill the menu captions, label and frame captions, tooltips, and so on, based on a set of ID numbers the Wizard generates.*

```
'set the controls' captions using
'the caption property for menu
'items and the Tag property
'for all other controls
For Each ctl In frm.Controls
   sCtlType = TypeName(ctl)
   If sCtlType = "Label" Then
      ctl.Caption = LoadResString(CInt(ctl.Tag))
   ElseIf sCtlType = "Menu" Then
      ctl.Caption = LoadResString(CInt(ctl.Caption))
   '(other controls here…)
   End If
Next
```

I've removed the code for the other controls from this example. For the complete code generated by the Wizard, go to the Registered Level of The Development Exchange (for details, see the Code Online box at the end of this column).

BROWSE THE INTERNET

You even can add a custom Web browser that jumps to your home page! The next step in the Application Wizard creates a browser form that uses the Web Browser control from the Microsoft Internet Controls (contained in the SHDOCVW.DLL) to add the same functionality of Microsoft Internet Explorer to your application. The code generated by the Wizard is easy to understand. Simply call the Navigate method of the control with a URL and add the URL to the combo box so the box displays the URL:

```
Private Sub Form_Load()
WebBrowser1.Navigate StartingAddress
cboAddress.Text = StartingAddress
End Sub
```

The next step in the Application Wizard gives you a choice of four standard forms to add to your application and lets you pick additional forms from your Templates\Forms folder. You can add a Splash screen at application startup, a Login dialog to accept a user ID and password, an Options dialog with four tabs for custom settings, and an About box.

Next, the Wizard creates data access forms from the tables and queries in a database. Although these forms use the Data control and include code to add, update, and delete records in the database, you probably want to add your own data validation routines.

Finally, give your project a name, sit back, and watch the Wizard build all the forms and generate standard code for your application. You can review a summary of the tasks the Wizard performed and get started on the To Do's. ☒

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.

Discover the Application Wizard

Locator+ Codes

Listings ZIP file, including the listings not printed here due to space limitations (free Registered Level): VBPJ0497
Complete listings for this article (subscriber Premier Level): GS0497P