The Keys to Your Controls

by Karl E. Peterson

Click & Retrieve Source CODE!

Q GENERIC ROUTINE ACTS ON CONTROL ARRAY

Is there any way to use a variable for a control name? One field in my recordset contains the names of menu items that need to be enabled or disabled at run time. I use a Select Case or If-Then-Else block to test all the possibilities:

```
set db as OpenDataBase("MyDB")
set rs as db.OpenRecordSet("MenuEnables")
Do While Not rs.Eof
   ' rs!mnuItem holds the name of Object
   ' in a menu list
   If rs!mnuItem = "mnuFileSave" Then
      mnuFileSave.Enabled = True
   ElseIf rs!mnuItem = "mnuFileExit" Then
      mnuFileExit.Enabled = True
   End If
   rs.MoveNext
Loop
```

However, there must be a better way.
—*David Codding, Bowling Green, Ohio*

A

The solution you've found was the best available in VB3, but the situation has much improved with later releases. VB4 introduced keyed collections, where you can access references to each item in a collection with a unique index key. Although VB3 offered the Controls collection, VB4 keyed this collection on the individual control names. You can put this feature to excellent use by rewriting your code along these lines:

```
Me.Controls(rs!mnuItem).Enabled = True
```

Of course, this technique works with any string variable:

```
Me.Controls("mnuFileSave").Enabled = True
```

You should be aware of one peculiarity of the Controls collection. Unlike other standard collections, the Controls collection returns two distinct types of objects. If the key (control name) refers to a control array, querying this Item results in an array reference. Using this trick, you can write a single generic routine that performs the same action on all elements of a control array. For example, use this routine to clear the contents of all text boxes in an array:

```
Sub ClearTextArray(Which As String, Frm As Form)
   Dim oArray As Object
   Dim ctl As Control
   On Error Resume Next
   Set oArray = Frm.Controls(Which)
   For Each ctl in oArray
      ctl.Text = ""
   Next ctl
End Sub
```

Clearing a form full of text boxes is simple:

```
Call ClearTextArray("Text1", Me)
```

Karl E. Peterson is a GIS analyst with a regional transportation planning agency and serves as a member of the Visual Basic Programmer's Journal *Technical Review and Editorial Advisory Boards. Based in Vancouver, Washington, he's also an independent programming consultant specializing in ActiveX controls and contributes to various journals. Karl coauthored* Visual Basic 4 How-To, *from Waite Group Press. Online, he's a Microsoft Developer MVP and a section leader in both of* VBPJ's *online forums. Contact Karl at karl@rtc.wa.gov.*

This is your forum for addressing the intricacies of Visual Basic. Send your questions, clever tips, and techniques. Visual Basic Programmer's Journal *will pay $25 for any submission, tip, or question we print. If your submission includes code, please send it electronically. Please include your mailing address with your submission. Mail submissions to Q&A Columnists, c/o Fawcette Technical Publications, 209 Hamilton Avenue, Palo Alto, CA, USA, 94301-2500. CompuServe: 74774,305. Internet: vbpjedit@fawcette.com.*

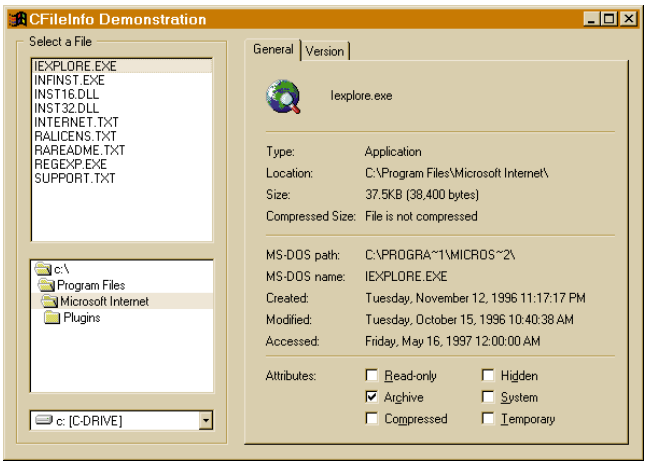

FIGURE 1 ***Two Classes Replicate Dialog.*** *This VB5 demo applet uses CFileInfo and CFileVersionInfo classes to replicate the Properties dialog used by NT4 and Windows 95. A single API call to FindFirstFile obtains most of the information shown on the General tab.*

Special thanks to Craig Clearman, a Microsoft Developer MVP, for bringing this capability to my attention.

Q DISPLAY PROPER-CASE FILE AND PATH NAMES

Using VB4/32 Professional Edition with Windows 95, I created a file picker form with the native FileListBox, DirListBox, DriveListBox, and TextBox controls. It works fairly well, except the file names and paths are all displayed and returned in lower case. I'd like to have the information displayed in the actual case. If I can't get it displayed in the proper case, how can I convert the lowercase name with proper-case?
—Richard Hendricks, Tweksbury, Massachusetts

A There's no way to alter the displays you see with the VB4 intrinsic controls. You might want to consider upgrading to VB5, which partially cured this annoyance (see Figure 1). You can use a simple API call, however, to correct path and file names with improper cases. In fact, you have a choice of two: either FindFirstFile or SHGetFileInfo. The safest pick is FindFirstFile, because NT 3.51 doesn't support SHGetFileInfo—it's one of the new shell calls. See just how simple it can be:

```
Function ProperCaseFile(FileIn As String) As String
   Dim hSearch As Long
   Dim wfd As WIN32_FIND_DATA
   hSearch = FindFirstFile (FileIn, wfd)
   If hSearch <> INVALID_HANDLE_VALUE Then
      Call FindClose(hSearch)
      ProperCaseFile = TrimNull (wfd.cFileName)
   End If
End Function
```

The ProperCaseFile function accepts a filespec, whose path may be a relative reference, and returns only the proper-case file name (see Listing 1, which also includes the TrimNull function). A lot of other, interesting data is also returned in the WIN32_FIND_DATA structure. With this one call, you can retrieve the file's attributes, size, and the short 8.3 file name, as well as its creation, last access, and last write time stamps.

You usually use FindFirstFile when you need to find all files that match a wildcard filespec. FindFirstFile opens a search handle and, as the name implies, returns information about the first file that matches. Once you establish this search handle, your program passes it to FindNextFile until all matching files are found. It's important, whether you're searching for one or many files, to always call FindClose on the search handle when you're done. Calling FindClose effectively cleans up some scratch buffers the system has allocated to track your search. Failing to call FindClose could cause a slow resource leak.

Proper-casing paths is a bit trickier. No single function will do it all with one call. However, you can use FindFirstFile to recursively work your way through each directory on a path, and solve the problem that way instead (see Listing 1).

If you've never worked with recursive functions, it will definitely help to step through this one a few times. ProperCasePath first whacks from the tail one directory at a time, calling itself and passing the truncated path, until it works its way back to the drive letter. At that point you can call FindFirstFile on each successive directory. As the function returns to itself, the results are appended back on, thus rebuilding the path.

To obtain the complete CFileInfo class, you can download it from the Registered Level of The Development Exchange (see the Code Online box at the end of this column for details). Additionally, subscribers to the Premier Level can download a companion class CFileVersionInfo and a demo applet that uses both classes to emulate the system Property Pages dialog used by NT4.

YOU USUALLY USE FINDFIRSTFILE WHEN YOU NEED TO FIND ALL FILES THAT MATCH A WILDCARD FILESPEC.

Q SOLVE FILE-LOADING PROBLEM

It seems that more and more of the project files sent to me by coworkers and collaborators refuse to load into the VB5 IDE. What's more, the forms lose controls—that is, a ListView control might be replaced with a PictureBox control. This is extremely irritating, as I must manually re-create the lost controls and reset all their design-time properties. What's going on?
—Zach Thomas, received by e-mail

A The answer is simple yet frustrating. Microsoft, yet again, has updated and made incompatible the Comctl32.ocx file that contains your missing ListViews. This time, the update is in fact "binary compatible," so older (but not too old!) programs should still run with the newer OCX. Problems do arise when developers (or machines) share source code with different versions. To immediately load the source, you can patch the project files. Open the VBP and FRM files in a text editor, and change the version number from 1.2 to 1.1. For example, change this line:

```
Object = "{6B7E6392-850A-101B-AFC0" & _
   "-4210102A8DA7}#1.2#0"; "comctl32.ocx"
```

to this:

```
Object = "{6B7E6392-850A-101B-AFC0" & _
   "-4210102A8DA7}#1.1#0"; "comctl32.ocx"
```

VB4 32-bit VB5

```
' Win32 API Declarations
Private Declare Function FindFirstFile _
  Lib "kernel32" Alias "FindFirstFileA" _
  (ByVal lpFileName As String, lpFindFileData _
  As WIN32_FIND_DATA) As Long
Private Declare Function FindClose Lib "kernel32" _
  (ByVal hFindFile As Long) As Long
' Win32 API Constants
Private Const MAX_PATH = 260
Private Const INVALID_HANDLE_VALUE = -1
Private Const FILE_ATTRIBUTE_DIRECTORY = &H10
' Win32 API Structures
Private Type WIN32_FIND_DATA
  dwFileAttributes As Long
  ftCreationTime As FILETIME
  ftLastAccessTime As FILETIME
  ftLastWriteTime As FILETIME
  nFileSizeHigh As Long
  nFileSizeLow As Long
  dwReserved0 As Long
  dwReserved1 As Long
  cFileName As String * MAX_PATH
  cAlternate As String * 14
End Type
Private Type FILETIME
  dwLowDateTime As Long
  dwHighDateTime As Long
End Type

Private Function ProperCasePath(ByVal PathIn As String) _
  As String
  Dim hSearch As Long
  Dim wfd As WIN32_FIND_DATA
  Dim PathOut As String
  Dim i As Long
  '
  ' Trim trailing backslash, unless root dir.
  '
  If Right(PathIn, 1) = "\" Then
     If Right(PathIn, 2) <> ":\" Then
        PathIn = Left(PathIn, Len(PathIn) - 1)
     Else
        ProperCasePath = UCase(PathIn)
        Exit Function
     End If
  End If
  '
  ' Insure that path portion of string uses the
  ' same case as the real pathname.
  '
  If InStr(PathIn, "\") Then
     For i = Len(PathIn) To 1 Step -1
        If Mid(PathIn, i, 1) = "\" Then
           '
           ' Found end of previous directory.
           ' Recurse back up into path.
           '
           PathOut = ProperCasePath(Left(PathIn, i - _
              1)) & "\"
           '
           ' Use FFF to proper-case current directory.
           '
           hSearch = FindFirstFile(PathIn, wfd)
           If hSearch <> INVALID_HANDLE_VALUE Then
              Call FindClose(hSearch)
              If wfd.dwFileAttributes And _
                 FILE_ATTRIBUTE_DIRECTORY Then
                 ProperCasePath = PathOut & _
                    TrimNull(wfd.cFileName)
              End If
           End If
           '
           ' Bail out of loop.
           '
           Exit For
        End If
     Next i
  Else
     '
     ' Just a drive letter and colon,
     ' upper-case and return.
     '
     ProperCasePath = UCase(PathIn)
  End If
End Function

Private Function TrimNull(ByVal StrIn As String) _
  As String
  Dim nul As Long
  '
  ' Truncate input string at first null.
  ' If no nulls, perform ordinary Trim.
  '
  nul = InStr(StrIn, vbNullChar)
  Select Case nul
     Case Is > 1
        TrimNull = Left(StrIn, nul - 1)
     Case 1
        TrimNull = ""
     Case 0
        TrimNull = Trim(StrIn)
  End Select
End Function
```

LISTING 1 ***Display Proper-Case Path Names.*** *Users notice the little things. If your application displays path names, perhaps in an Options dialog that points to a data source, it's a nice touch to ensure that you display them using the proper case. The ProperCasePath function recursively works its way through each directory in a path string.*

This allows you to use projects designed with the newer control on systems with the older control. The long-term solution is to download and install the latest build. It's currently shipping with a number of products, which explains why you're seeing references to it arrive more often. At deadline, version 5.00.3828 is the latest, and is available from Microsoft Knowledge Base article Q167121 (http://www.microsoft.com/kb/articles/q167/1/21.htm).

Microsoft also shipped two other versions of Comctl32.ocx, which are binary incompatible with those that shipped most recently. If you have loaded either Beta-1 of the VB5 Control Creation Edition (5.00.3422) or the Microsoft Office 97 Developer Edition (5.00.3601), you should confirm that these older versions are no longer on your machine. Version 5.00.3714 shipped with the retail release of VB5. For more information, see KB article Q167123 (http://www.microsoft.com/kb/articles/q167/1/23.htm). ×

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.

The Keys to Your Controls

Locator+ Codes

Listings ZIP file plus the CFileInfo class (free Registered Level): VBPJ0897
✪ Listings for this article, plus a companion class CFileVersionInfo and a demo applet that uses both classes to emulate the system Property Pages dialog used by NT4 (subscriber Premier Level): QA0897P