Taking the IIF Approach to Nulls

Click & Retrieve Source CODE!

by Carl Franklin

Q NULL IIFs, ANDS, OR BUTS ABOUT IT

One of the user tips in the July 1995 issue of *VBPJ* addressed properly dealing with nulls. Opting for the "do the most with the least code" option, I use an approach that places the value "N/A" into Value$ if the string is null. Otherwise it uses the data itself:

```
Value$ = IIF(IsNull(MyDynaset!ClientNumber), _
   "N/A", MyDynaset!ClientNumber)
```

This provides good control over what happens if the value is null, and is also more efficient than an If...Then...Else construct.
—*J. Boyd Nolan, received by e-mail*

A

This solution works well. The only thing you need to be careful about is that if you use IIF in VB 3.0, you must ship MSAFINX.DLL along with your executable because IIF is located in this DLL.

For those who are not familiar with IIF (Immediate IF), it returns one of two parts depending on the evaluation of an expression. For example:

```
ReturnValue = IIf(Expression, Truepart, _
   Falsepart)
```

Expression is any valid expression such as (n = 1). Truepart is a constant, variable, or other member that is returned if the expression is true. Falsepart is a constant, variable, or other member that is returned if the expression is false. So, this code will always return "OK":

```
ReturnValue$ = IIf((2 > 1), "OK", "Error")
```

Carl Franklin is a software developer and the co-owner of Carl & Gary's Visual Basic Home Page (http://www.apexsc.com/vb). Carl is also the author of Visual Basic Internet Programming, *available from John Wiley & Sons. In his spare time, Carl is researching a cure for feline pattern baldness. Reach Carl by e-mail at carlf@apexsc.com or through* Visual Basic Programmer's Journal. *CompuServe users, address your mail to internet:carlf@apexsc.com.*

This is your forum for addressing the intricacies of the Visual Basic language. 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 a disk along with your hard copy. Mail submissions to Q&A Columnists, c/o Fawcette Technical Publications, 209 Hamilton Avenue, Palo Alto, CA, USA, 94301-2500. CompuServe: 74774,305.*

VB4

```
Option Explicit

Const WM_SYSCOMMAND =3D &H112&

Const SC_CLOSE =3D &HF060&
Const SC_MAXIMIZE =3D &HF030&
Const SC_MINIMIZE =3D &HF020&
Const SC_MOVE =3D &HF012&
Const SC_RESTORE =3D &HF120&
Const SC_SIZE =3D &HF008&

Private Sub Form_Load()

  '-- Trap messages for this form
  MsgBlaster1.hWndTarget =3D Me.hWnd

  '-- Trap the system command message
  MsgBlaster1.AddMessage WM_SYSCOMMAND, 1

End Sub

Private Sub MsgBlaster1_Message(ByVal hWnd As Long, _
  ByVal Msg As Long, =
wParam As Long, lParam As Long, nPassage As Integer, _
  lReturnValue As =
Long)

  Dim szMsg   As String
     =20
  Select Case Msg
     =20
     Case WM_SYSCOMMAND
     =20
        Select Case wParam
           '-- Create a descriptive message based
           '  on the system command message
           Case SC_CLOSE
              szMsg =3D "Closing Window"
           Case SC_MAXIMIZE
              szMsg =3D "Maximizing Window"
           Case SC_MINIMIZE
              szMsg =3D "Minimizing Window"
           Case SC_MOVE
              szMsg =3D "Moving Window"
           Case SC_RESTORE
              szMsg =3D "Restoring Window"
           Case SC_SIZE
              szMsg =3D "Sizing Window"
        End Select
  =20
        '-- Display the descriptive message
        '  in the form's caption
        Caption =3D szMsg
  =20
  End Select
  =20
End Sub
```

LISTING 1 ***I Can Name That Action in a Descriptive Message.*** *This program, which consists of a single form with a Message Blaster control on it, traps some of the WM_SYSCOMMAND Messages. As you move, size, minimize, maximize, restore, or close the form, the caption will change to display a descriptive message of your actions.*

And this will always return "Error":

```
ReturnValue$ = IIf((1 > 2), "OK", _
   "Error")
```

This will return "OK" if n is greater than three, otherwise it returns "Error":

```
ReturnValue$ = IIf((n > 3), "OK", _
   "Error")
```

Q GETTING TO THE ROOT OF IT

I created a database program in VB 3.0 that works as expected on my own computer. The problem arises when I create an EXE file and migrate the program to a friend's machine. Both of us are running Windows 95 and, after I have installed the EXE file with accompanying VBXs, DLLs, and so forth, and then double-click on the Apps icon, an error, "Invalid Property Value," is generated. To be honest, I'm lost! The error is not generated in my VB program and, if I run the EXE file on my machine, no such error occurs. I am using the Eschalon installation utility, which I find to be excellent. Do you have any ideas as to what might be causing this bug?
—Michael Griffiths, received by e-mail

A Invalid Property Value occurs when you set a property to an invalid value, such as setting Form1.Top = "Hello There!"

The problem is either with a control or the actual value. If it's the control, then there is probably an older version of a VBX

VB4

```
Open MyFile$ For Binary As 1

NumBlocks% = Lof(1) \ 8192
Remainder% = Lof(1) Mod 8192

For i% = 1 To NumBlocks%
  Buffer$ = Space$(8192)
  Get #1, , Buffer$

  '-- Process Buffer$ here..
  '  8K worth of data

Next

If Remainder% Then
  '-- Last block =< 8192
  Buffer$ = Space$(Remainder%)
  Get #1, , Buffer$

  '-- Process last block here.

End If

Close 1
```

LISTING 2 ***Binary Mode Speeds Access.*** *You can really speed up access to text files by using Binary mode, reading in larger chunks of data, and then processing that data in memory using string functions.*

on your friend's machine, and you didn't properly install the correct version. If you know you installed the correct version, a quick reboot will tell you if it was caused by an older version in memory.

If it's the value, then you have to do a little debugging. If you can't install VB on your friend's machine, then you need to put what I call "markers" in your app, such as writing to a file or popping a message box. I prefer the latter because you can immediately tell where the program is without having to walk through a file. Try to find the statement where the error occurs in this way. Although you'll probably have to build several EXEs, it's the only way to solve the problem at this point. Of course, the whole time you're doing this, you should be thinking of possible gotchas in the code leading up to the error.

Q ACT LIKE A TAB, PLEASE

As a newcomer to Visual Basic, I'm trying to customize the TextBox control in VB4 (16-bit). I want a carriage return to act as a tab, without hard-coding it. In other words, I would like Enter to take me to a specific control, not the next one in the Tab Order.

Also, I want to automatically add dashes in a phone number entry field. When I changed TextBox.Text, it automatically redrew the field and moved the cursor to the first character. Is there a way I can customize TextBox to do this?

—Tim Ryder, received by e-mail

A You can make a carriage return act just like a tab with this line of code in your text box control's KeyPress event:

```
Private SubText1 KeyPress_
   (KeyAscii As Integer)

If KeyAscii = 13 Then
   SendKeys "{TAB}"
End If

End Sub
```

The SendKeys statement sends one or more keystrokes to the window currently in focus, just as if you had pressed them yourself. Check out SendKeys in the help file for more information.

As far as adding dashes automatically to the text box, my first suggestion would be to use the masked text control, which gives you more control over this type of action.

However, you can still append text to the end of a line of text without redrawing the entire line by using the SelStart and SelText properties like this:

```
Sub AppendText (Text1 As Control, _
   szText As String)

   Text1.SelStart = Len(Text1) + 1
   Text1.SelText = szText

End Sub
```

SelStart sets the position of the Text control's internal cursor, and SelText inserts the text at the current position.

Q SET AN OCX TRAP

I am interested in trapping win dows messages in Vi-sual Basic 4.0 (32-bit). Is there an OCX that traps windows messages in a similar fashion as the MSGBLAST.VBX?

—Ulf Erik Forsbakk, received by e-mail

A WareWithAll Message Blaster is a 32-bit OCX with a nice interface comparable to the publicly available MSGBLAST.VBX. You can download a trial version of this OCX from Carl & Gary's Visual Basic Home Page: http://www.apexsc.com/vb/ftp/misc/mblast32.zip.

It works exactly like the VBX. You give it an hWnd for the window (form) you want to trap messages from, and then you give it the message numbers you wish to trap. It fires an event upon the receipt of those messages.

I created a small example that traps some of the WM_SYSCOMMAND Messages (see Listing 1). The program consists of a single form with a Message Blaster control on it. As you move, size, minimize, maximize, restore, or close the form, the caption will change to display a descriptive message of your actions.

Q TAKE BIGGER BYTES

I am working on an application that needs to read in text files that are, for the most part, loosely structured. The first line can be broken down into columns, but the remainder of the file is free text of unspecified lengths. Although I have no problems reading in the files, the process is very slow. Reading three of these files for a total of 15,647 bytes takes just over six seconds using Access 2.0 macros (this also includes putting the data into Access tables). It takes just over 26 seconds to read the same three files using VB 4.0 32-bit, without putting the data into a database. I am reading the text files byte by byte once I get past the structured first line, by using GET. I have played with the LEN= variable on the OPEN statement, but it just got worse.

Is there a way to read each file in a different operating thread? Or is there something else I can do to speed this up in VB4?

—Paul Little, Encinitas, California

A If you are talking about standard text files here—each line ending with a Cr/Lf—you can read in one line at a time using this approach:

```
Open MyFile$ For Input As 1

Do Until Eof(1)
   Line Input #1, NextLine$

   '— Process one line of text here

Loop

Close 1
```

I don't know why Access was so forgiving with your data. As I see it, the problem with your code is that you are reading one byte at a time when you could be reading a line at a time or, in general, more data per disk access.

You can speed file access by reading in 8K chunks (or larger) in binary mode, and then (if you have to) walk through the data in memory (see Listing 2).

The more data you read from the disk in one shot, the less time it will take to read it. Walking through a large amount of data in memory one byte at a time is also time consuming, but you can use Instr and other string functions to manipulate strings. ×