

Re: Unicode Button in VB6

Source: <http://www.tech-archive.net/Archive/VB/microsoft.public.vb.winapi/2006-04/msg00123.html>

- *From:* "J r mie Gent" <jer_kjr@xxxxxxxxxxxxx>
 - *Date:* Sat, 15 Apr 2006 19:24:38 +0200
-

Sorry, you should read this
Set ts = fs.OpenTextFile("c:\Greek.txt", , , -1)
at the line where I open the textfile (for the file to be considered as
unicode)
Still, it doesn't work.
J r mie

"J r mie Gent" <jer_kjr@xxxxxxxxxxxxx> wrote in message
news:O228%230KYGHA.4920@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Thanks so much Thorsten!
Your explanation is really clear, and now I understand better this concept
of suclassing (I found also another clear explanation here:
<http://www.geocities.com/SiliconValley/Lab/1632/atch12.html>)
So with this, I tried Timo's trick with the DefWindowProcW call
(<http://www.eggheadcafe.com/ng/microsoft.public.vb.controls/post252068.asp>)

I made this:

In the module code:

```
Public Const GWL_WNDPROC = -4  
Public Const WM_SETTEXT = &HC
```

```
Public Declare Function SetWindowLong Lib "user32" Alias "SetWindowLongA"  
(ByVal hWnd As Long, ByVal nIndex As Long, ByVal dwNewLong As Long) As  
Long
```

```
Public Declare Function CallWindowProc Lib "user32" Alias  
"CallWindowProcA" (ByVal lpPrevWndFunc As Long, ByVal hWnd As Long, ByVal  
Msg As Long, ByVal wParam As Long, ByVal lParam As Long) As Long
```

```
Private Declare Function DefWindowProc Lib "user32" Alias "DefWindowProcW"  
(ByVal hWnd As Long, ByVal wParam As Long, ByVal lParam As Long, ByVal  
lParam As Long) As Long
```

```
Public oldWndProc As Long
```

```
Public Function MyWndProc(ByVal hWnd As Long, ByVal wParam As Long, ByVal  
wParam As Long, ByVal lParam As Long) As Long
```

Re: Unicode Button in VB6

```
'Debug.Print wMsg & " " & wParam & " " & lParam
If wMsg <> WM_SETTEXT Then
MyWndProc = CallWindowProc(oldWndProc, hWnd, wMsg, wParam, lParam)
Else
MyWndProc = 0
End If
End Function
```

```
Public Sub SetUnicodeWindowText(ByVal hWnd As Long, ByVal NewText As
String)
DefWindowProc hWnd, WM_SETTEXT, &H0&, StrPtr(NewText)
End Sub
```

In the form (containing a simple command button):

```
Private Sub Form_Load()
Dim fs As New FileSystemObject, ts As TextStream, myString As String
Set ts = fs.OpenTextFile("c:\Greek.txt")
myString = ts.ReadAll: ts.Close: Set ts = Nothing: Set fs = Nothing
```

```
oldWndProc = SetWindowLong(Me.Command1.hWnd, GWL_WNDPROC, AddressOf
MyWndProc)
Call SetUnicodeWindowText(Me.Command1.hWnd, myString)
End Sub
```

```
Private Sub Form_Unload(Cancel As Integer)
Ret& = SetWindowLong(Me.Command1.hWnd, GWL_WNDPROC, oldWndProc)
End Sub
```

The file c:\Greek.txt is unicode and contains a single word written in greek (appearing correctly in excel for example)
I changed the call to SetUnicodeText (I removed the VarPtr), otherwise the adress of the string is rendered (double call with StrPtr in the function)
But as Timo saied, unfortunately it doesn't work.

That's really too bad, because this would make the whole thing so easy.
There must definitely be a way though, do you have any further idea ? (I looked in Button Default Message Processing as you advised, but it's not what I need, unless you mean I really have to handle the WM_PAINT and paint the characters myself ... Which would seem to me quite complicated).

Again, guys, your comments are very helpful, so thank you very much!

Jérémie

PS Thorsten: I hope the weather will be better tomorrow in Freiburg, it's

Re: Unicode Button in VB6

kindof bad for an Easter week-end, oder ?

"Thorsten Albers" <albersRE@xxxxxxxxxxxxxxxxxxxx> wrote in message [news:01c660a1\\$efa050a0\\$b552f8d9@xxxxxxxxxxxx](mailto:news:01c660a1$efa050a0$b552f8d9@xxxxxxxxxxxx)

J r mie Gent <jer_kjr@xxxxxxxxxxxx> schrieb im Beitrag <#xMO6S0XGHA.4768@xxxxxxxxxxxxxxxxxxxxxxxx>...

That looks good to me, but isn't detailed enough for me. I don't exactly know how to subclass my window to avoid to conversion Unicode -> Ansi. I suppose the call to SetWindowLongW might need a special GWL_EXSTYLE, but

I

don't know which and how.

You are subclassing a window by replacing the original window procedure with your own window procedure. Therefore to subclass a window you have to call SetWindowLong() with GWL_WNDPROC as the second and the address of the new window procedure as the third parameter. There is no need to call SetWindowLongW() because the VB standard controls are all of ANSI window classes and not UNICODE window classes. Your own window procedure has to be in a VB standard module (*.BAS), there is no other place where you can put it. You retrieve the address of the window procedure with the VB statement AddressOf. E.g.:

----- BAS module:

```
Public Function MyWindowProc _
(
  ByVal hWnd As Long, _
  ByVal lParam As Long, _
  ByVal wParam As Long, _
  ByVal lParam As Long _
) As Long
...
End Function
```

----- Form module:

```
pfnWindowProcPrev = SetWindowLong(...hWnd, GWL_WNDPROC,
```

Re: Unicode Button in VB6

```
AddressOf  
MyWindowProc)  
If pfnWindowProcPrev = 0 Then  
' ... Failure  
End If
```

You have to save the value returned by `SetWindowLong()` which is the address of the previous window procedure. This address is needed for two reasons:

1. If your application is about to finish, it has to restore the original window procedure addresses of all subclassed windows (i.e. 'unsubclass' them). A place where to do this is the `QueryUnload` or `Unload` event.
2. Messages not handled by your own window procedure have to be passed to the original window procedure (`CallWindowProc(pfnWindowProcPrev, ...)`).

By this it is obvious that `pfnWindowProcPrev` has to be accessible both in the form module and the BAS module. A good place to store it therefore is in a window property of the subclassed window (`SetProp()`, `GetProp()`, `RemoveProp()`).

After you have successfully subclassed the button window your own window procedure has to process all window messages that are sent to the button window in order to (re)paint the button text. To decide how to handle messages a good approach is reading the chapter "Button Default Message Processing" in the MSDN. The most important message to be handled by your own window procedure is of course `WM_PAINT`.

When subclassing a VB standard control remember that these controls themselves are already subclassed controls (the VB standard controls do not subclass the window but the respective standard window class), so their behaviour may be different from the standard window class they are using.

--

THORSTEN ALBERS Universität Freiburg
albers@
uni-freiburg.de
