

A "NEW! IMPROVED!" wscript.sleep...

Source:

<http://www.tech-archive.net/Archive/Scripting/microsoft.public.scripting.vbscript/2005-07/msg00735.html>

- *From:* mr_unreliable <kindlyReplyToNewsgroup@xxxxxxxxxxxx>
 - *Date:* Wed, 20 Jul 2005 11:51:33 -0400
-

```
' --- the "NEW! IMPROVED!" WScript.Sleep Method -----
' Back in the days when I was young-and-innocent, I imagined that
'   "wscript.sleep" was just a scripting wrapper for the sleep api,
'   and the original version of this reconstructed "wscript.sleep"
'   version reflected that, i.e.:
'       apiSleep CLng(tSnooze) ' call the sleep api...
'
' 20July05: but now, I'm older-and-wiser (well, older anyway).
'   And, as has been brought to my attention, wscript.sleep is NOT
'   the same as api sleep, it's much more complicated. Although,
'   for very short "sleeps", it doesn't make much difference.
'
'   But for longer sleeps, there is a danger of the app becoming
'   "non-responsive". That situation arises because the system
'   can send out messages to a specific app (say, to kill it),
'   or can send out messages to all apps (called broadcasting)
'   (say, when it wants to shutdown). Any app that doesn't respond
'   to these system messages is declared "non-responsive". And, if
'   you are responsible for a non-responsive app, then you better
'   not show your face in Redmond, or else you will have Microsoft
'   scorn heaped upon you, as you are not conforming with Microsoft's
'   basic (no pun intended) programming standards.
'
'   Note that if you are in "api sleep", your app won't respond to
'   those system messages, but if you are in "wscript.sleep" then
'   your script WILL respond. Try it. Put your script in a 10 minute
'   sleep, and try killing your script with the "close program"
'   (ctrl-alt-del) dialog. Your script will respond and
'   terminate itself.
'
'   And so, how does Microsoft do that??? (Secret Disclosure:)
'   Microsoft doesn't really go to sleep in wscript.sleep for any
'   substantial length of time. Microsoft goes into a loop, and
'   sleeps for a (short) while, then wakes up and checks for any
'   system messages that may be intended for it. If there are any
'   (messages) it will process those messages, and then goes back
'   to sleep again. If there are no messages, it just goes back
'   to sleep...
'
'   O.K., so that was a little "long-winded", but it does explain
'   what the NEW! IMPROVED! wscript.sleep method below is all about.
' --- end of discussion -----
```

cheers, jw

A "NEW! IMPROVED!" wscript.sleep...

You got questions? WE GOT ANSWERS!!! ..(but,
no guarantee the answers will be applicable to the questions)

VERSION 1.0 CLASS

BEGIN

MultiUse = -1 'True

END

Attribute VB_Name = "clsWScript"

Attribute VB_GlobalNameSpace = False

Attribute VB_Creatable = True

Attribute VB_PredeclaredId = False

Attribute VB_Exposed = False

' WScript Class, an attempt to provide WScript Functionality

' to scripts running in the ms ScriptControl, jw 09Jun01

Option Explicit

' ---- Declarations and Constants -----

Private Declare Sub apiSleep Lib "kernel32" Alias "Sleep" (ByVal dwMilliseconds As Long)

,

Private Declare Function SetCurrentDirectory Lib "kernel32" _

Alias "SetCurrentDirectoryA" (ByVal lpPathName As String) As Long

Private Declare Function GetCurrentDirectory Lib "kernel32" _

Alias "GetCurrentDirectoryA" (ByVal nBufferLength As Long, ByVal lpBuffer As String) As Long

,

Private Const MAX_PATH = 260

,

Private m_myParent As frmMain ' parent of this class as form object

,

Private nRtn As Long ' used to hold api return values...

' ---- end of declarations and constants -----

,

' ---- Let/Get Parent -----

' the following two routines are part of the "callback" process, whereby the
' parent (control) and the child (this class) exchange calling cards.

' The "callback" is used as a way to access code and objects in the parent form.

' For more info on this, get yourself a copy of Dan Appleman's book, entitled:

' "Developing ActiveX Components w/vb 5", pg 243.

' -----

Public Property Let Parent(pSource As frmMain)

Debug.Print "[myForm] .. frmSetParent called"

' debugging stuff (bugassert expects TRUE)...

ArgAssert Not (pSource Is Nothing), "[LetParent], pSource is nothing!!!"

Set m_myParent = pSource

A "NEW! IMPROVED!" wscript.sleep...

A "NEW! IMPROVED!" wscript.sleep...

End Property

Public Property Get Parent() As frmMain

Debug.Print "[myForm] .. frmGetParent called"

' debugging stuff (bugassert expects TRUE)...

ArgAssert Not (m_myParent Is Nothing), "[GetParent], m_myParent is nothing!!!"

Set Parent = m_myParent

End Property

' ---- end of callback stuff -----

Public Property Get CreateObject(vProgID As Variant, Optional vPrefix As Variant = "") As Object

Set CreateObject = vbCreateObject(CStr(vProgID))

' if there is a second argument provided, then connect it...

' If IsMissing(vPrefix) Then Exit Property

' Parent.wsConnect CStr(vPrefix), CreateObject

End Property

' ---- Get / Set Current Working Directory -----

Public Property Let CurrentDirectory(vRHS As Variant)

nRtn = SetCurrentDirectory(CStr(vRHS))

End Property

Public Property Get CurrentDirectory() As Variant

Dim sBuf As String

Dim iNull As Integer

sBuf = String(MAX_PATH, Chr(0)) ' allocate string buffer

nRtn = GetCurrentDirectory(MAX_PATH, sBuf)

iNull = InStr(sBuf, Chr(0)) ' trim the returned string...

If iNull > 0 Then sBuf = Left(sBuf, iNull)

CurrentDirectory = sBuf

End Property

' ---- "Easy" WScript Methods (Echo, Sleep, Quit) ----

Public Sub Echo(sMsg As Variant)

' ignore any multiple-string-parameters, for now...

MsgBox CStr(sMsg)

End Sub

' ---- the "NEW! IMPROVED!" WScript.Sleep Method -----

' Back in the days when I was young-and-innocent, I imagined that

' "wscript.sleep" was just a scripting wrapper for the sleep api,

A "NEW! IMPROVED!" wscript.sleep...

A "NEW! IMPROVED!" wscript.sleep...

```
' and the original version of this reconstructed "wscript.sleep"
' version reflected that, i.e.:
' apiSleep CLng(tSnooze) ' call the sleep api...
'
' 20July05: but now, I'm older-and-wiser (well, older anyway).
' And, as has been brought to my attention, wscript.sleep is NOT
' the same as api sleep, it's much more complicated. Although,
' for very short "sleeps", it doesn't make much difference.
'
' But for longer sleeps, there is a danger of the app becoming
' "non-responsive". That situation arises because the system
' can send out messages to a specific app (say, to kill it),
' or can send out messages to all apps (called broadcasting)
' (say, when it wants to shutdown). Any app that doesn't respond
' to these system messages is declared "non-responsive". And, if
' you are responsible for a non-responsive app, then you better
' not show your face in Redmond, or else you will have Microsoft
' scorn heaped upon you, as you are not conforming with Microsoft's
' basic (no pun intended) programming standards.
'
' Note that if you are in "api sleep", your app won't respond to
' those system messages, but if you are in "wscript.sleep" then
' your script WILL respond. Try it. Put your script in a 10 minute
' sleep, and try killing your script with the "close program"
' (ctrl-alt-del) dialog. Your script will respond and
' terminate itself.
'
' And so, how does Microsoft do that??? (Secret Disclosure:)
' Microsoft doesn't really go to sleep in wscript.sleep for any
' substantial length of time. Microsoft goes into a loop, and
' sleeps for a (short) while, then wakes up and checks for any
' system messages that may be intended for it. If there are any
' (messages) it will process those messages, and then goes back
' to sleep again. If there are no messages, it just goes back
' to sleep...
'
' O.K., so that was a little "long-winded", but it does explain
' what the NEW! IMPROVED! wscript.sleep method below is all about.
' --- end of discussion -----
```

```
Public Sub Sleep(tSnooze As Variant)
Const tSleepInterval As Long = 200 ' ms
Dim lngSnooze As Long
Dim tElapsed As Long

lngSnooze = CLng(tSnooze) ' type-cast as long

If (lngSnooze < tSleepInterval) Then

apiSleep lngSnooze ' call the sleep api...
Exit Sub ' we're done here...

A "NEW! IMPROVED!" wscript.sleep...
```

```
Else ' for longer sleeps, allow for processing events...

For tElapsed = 0 To lngSnooze Step tSleepInterval

apiSleep tSleepInterval ' take a "short sleep"...
DoEvents ' allow for processing events...
' Debug.Print "[WScrips.Sleep], tElapsed: " & CStr(tElapsed)

Next ' tElapsed interval

End If ' interval test
End Sub

Public Sub Quit()
Parent.QuitScript
End Sub
```

- **References:**

- ◆ **[Running a script using VB6 ScriptControl](#)**

- ◇ *From:* Ron Lessnick

- ◆ **[Re: Running a script using VB6 ScriptControl](#)**

- ◇ *From:* mr_unreliable

- Prev by Date: **[Re: List OU's in an OU using VBS](#)**
- Next by Date: **[change mouse to hourglass?](#)**
- Previous by thread: **[Re: Running a script using VB6 ScriptControl](#)**
- Next by thread: **[WE INVITE YOU](#)**
- Index(es):
 - ◆ **[Date](#)**
 - ◆ **[Thread](#)**