

[Home Page](#) [Novità](#) [Aiuto](#)

Avviare un programma esterno ed attenderne la chiusura

http://www.vbsimple.net/howto/ht_062.htm

Difficoltà: 2 / 5

Abbiamo già visto [in un altro articolo](#) come avviare un programma esterno utilizzando la funzione **Shell**; tuttavia quella soluzione in molti casi non si rivela la scelta migliore; infatti la funzione Shell avvia il programma e immediatamente ritorna al progetto VB in modo da continuare l'esecuzione in maniera sincrona al processo esterno in esecuzione.

In molti casi invece è necessario avviare un processo esterno ed attendere finché questo non sia terminato; per far ciò la funzione Shell si rivela del tutto inutile ed è necessario procedere in un'altra maniera. Vedremo in questo articolo come sviluppare la cosiddetta Shell sospensiva che, per comodità, racchiuderemo all'interno di un unico modulo standard.

Inizieremo proprio da questo modulo e dal codice vero e proprio e solo dopo passeremo alla dimostrazione del funzionamento:

```
1. Option Explicit
2.
3. Private Type STARTUPINFO
4.     cb As Long
5.     lpReserved As Long
6.     lpDesktop As Long
7.     lpTitle As Long
8.     dwX As Long
9.     dwY As Long
10.    dwXSize As Long
11.    dwYSize As Long
12.    dwXCountChars As Long
13.    dwYCountChars As Long
14.    dwFillAttribute As Long
15.    dwFlags As Long
16.    wShowWindow As Integer
17.    cbReserved2 As Integer
18.    lpReserved2 As Long
19.    hStdInput As Long
20.    hStdOutput As Long
21.    hStdError As Long
22. End Type
23.
```

La struttura **STARTUPINFO** verrà utilizzata per specificare le opzioni sul processo esterno da avviare; per ragioni di completezza riportiamo la descrizione di tutti i campi poiché possono rendersi utili in situazioni differenti da quella riportata: **cb** indicherà la lunghezza in bytes della struttura e rappresenta un valore obbligatorio; **lpReserved** indica un valore riservato e deve essere 0; **lpDesktop** in un ambiente multidesktop indicherà il nome del desktop sul quale visualizzare la nuova applicazione; **lpTitle** (valido solo per le applicazioni console) specifica il titolo mostrato sulla finestra della console; **dwX** e **dwY** indicano la coordinata della nuova finestra da mostrare; **dwXSize** e **dwYSize** indicheranno

la larghezza e l'altezza della finestra da visualizzare; *dwXCountChars* e *dwYCountChars* (validi solo per le applicazioni console) specificheranno il numero di caratteri per riga e per colonna nella nuova finestra console; *dwFillAttribute* (valido solo per le applicazioni console) consente di specificare la combinazione di colori di testo e di sfondo; *dwFlags* comanda l'uso degli altri campi e può contenere una combinazione qualsiasi dei valori riportati di seguito; *wShowWindow* consente la specifica della modalità di visualizzazione della nuova finestra (corrisponde ai valori dell'enumerazione  *VbAppWinStyle*); *cbReserved2* e *lpReserved2* rappresentano valori riservati e non andrebbero specificati; infine *hStdInput*, *hStdOutput* e *hStdError* consentono di specificare i 3 [handle](#) per lo *Standard Input*, *Standard Output* e *Standard Error*.

Il valore del campo *dwFlags* può combinare una o più delle seguenti costanti:

- `STARTF_USESHOWWINDOW (&H1)`
Specifica l'uso del valore nel campo *wShowWindow*;
- `STARTF_USESIZE (&H2)`
Specifica l'uso dei campi *dwXSize* e *dwYSize*;
- `STARTF_USEPOSITION (&H4)`
Specifica l'uso dei campi *dwX* e *dwY*;
- `STARTF_USECOUNTCHARS (&H8)`
Specifica l'uso dei campi *dwXCountChars* e *dwYCountChars*;
- `STARTF_USEFILLATTRIBUTE (&H10)`
Specifica l'uso del campo *dwFillAttribute*;
- `STARTF_RUNFULLSCREEN (&H20)`
Specifica l'avvio dell'applicazione console a pieno schermo;
- `STARTF_FORCEONFEEDBACK (&H40)`
Attiva forzatamente il cursore di attesa durante l'avvio dell'applicazione;
- `STARTF_FORCEOFFFEEDBACK (&H80)`
Disattiva forzatamente il cursore di attesa durante l'avvio;
- `STARTF_USESTDHANDLES (&H100)`
Specifica l'uso dei campi *hStdInput*, *hStdOutput* e *hStdError*.



Più di una documentazione riporta un'erronea rappresentazione del campo *lpReserved2* all'interno della struttura *STARTUPINFO*; l'errore nasce da una cattiva traduzione della definizione C della struttura stessa che riporta `LPBYTE lpReserved2` (`LPBYTE` indica Long Pointer Byte cioè un puntatore ad un array di bytes e quindi un normale puntatore a 32 bit) e che è stato a volte riportato come `lpReserved2 As Byte`. Al di là del fatto che funzioni comunque, a causa dell'allineamento ai 32 bit effettuato da Visual Basic, la struttura riportata in questo articolo è tuttavia quella corretta.

```

24. Private Type PROCESS_INFORMATION
25.     hProcess As Long
26.     hThread As Long
27.     dwProcessId As Long
28.     dwThreadId As Long
29. End Type
30.
31. Private Declare Function CreateProcess Lib "kernel32" Alias
    "CreateProcessA" (ByVal lpApplicationName As Long, ByVal lpCommandLine As
    String, ByVal lpProcessAttributes As Long, ByVal lpThreadAttributes As
    Long, ByVal bInheritHandles As Long, ByVal dwCreationFlags As Long, ByVal
    lpEnvironment As Long, ByVal lpCurrentDirectory As String, lpStartupInfo As

```

```

    STARTUPINFO, lpProcessInformation As PROCESS_INFORMATION) As Long
32. Private Declare Function WaitForSingleObject Lib "kernel32" (ByVal hHandle
    As Long, ByVal dwMilliseconds As Long) As Long
33. Private Declare Function CloseHandle Lib "kernel32" (ByVal hObject As Long)
    As Long
34.
35. Public Const WAIT_TIMEOUT As Long = 258&
36. Private Const WAIT_FAILED As Long = &HFFFFFFFF
37. Private Const NORMAL_PRIORITY_CLASS = &H20&
38.

```

La struttura **PROCESS_INFORMATION** è utilizzata solo per contenere gli [handle](#) e i numeri identificativi del [processo e del thread](#) eseguiti; seguono le dichiarazioni delle tre funzioni utilizzate in seguito: **CreateProcess** si occuperà della reale apertura del processo esterno e richiede fra l'altro la specifica della cartella da cui avviare l'applicazione e della priorità da assegnare al nuovo processo. **WaitForSingleObject** è una funzione generica utilizzata solitamente per altri scopi ed in questo caso sfruttata per forzare l'attesa fintanto che il processo esterno non venga chiuso oppure per il periodo di tempo indicato. **CloseHandle** infine serve solamente per chiudere un handle aperto.

La costante **WAIT_TIMEOUT** corrisponde al valore di ritorno di WaitForSingleObject in caso di supero del limite di tempo di attesa massimo; **WAIT_FAILED** corrisponde invece alla situazione di errore durante l'attesa; la costante **NORMAL_PRIORITY_CLASS** rappresenta la priorità normale e verrà indicata nella chiamata alla funzione **CreateProcess**. Sono state omesse per brevità le dichiarazioni delle costanti **STARTF** indicate in precedenza.

```

39. Public Function ShellSospensiva(ByVal CommandLine As String, ByVal
    Directory As String, ByVal WindowStyle As VbAppWinStyle, ByVal WaitTime As
    Long) As Long
40.     Dim proc As PROCESS_INFORMATION
41.     Dim Start As STARTUPINFO
42.     With Start
43.         .dwFlags = STARTF_USESHOWWINDOW
44.         .wShowWindow = WindowStyle
45.         .cb = Len(Start)
46.     End With
47.     Call CreateProcess(0, CommandLine, 0, 0, 0, NORMAL_PRIORITY_CLASS, 0,
    Directory, Start, proc)
48.     If proc.hProcess <> 0 Then
49.         ShellSospensiva = WaitForSingleObject(proc.hProcess, WaitTime)
50.         Call CloseHandle(proc.hProcess)
51.     Else
52.         ShellSospensiva = WAIT_FAILED
53.     End If
54. End Sub

```

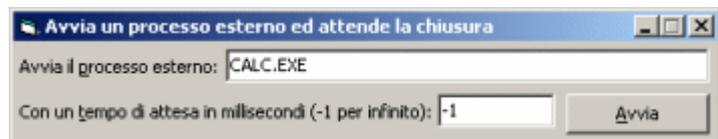
L'intera funzione **ShellSospensiva** si compone di poche e semplici righe e richiede la specifica del processo da eseguire, della cartella da cui avviarlo, dello stile della finestra da visualizzare e del tempo massimo per l'attesa in millisecondi; è possibile specificare un tempo di attesa infinito semplicemente fornendo il valore -1 (corrispondente al valore della costante **INFINITE** &HFFFFFFFF).

Inizializzata la variabile **Start** con la specifica della modalità di visualizzazione e lunghezza della struttura (righe 42-46) potremo procedere al richiamo del processo esterno indicato dal parametro **CommandLine** eseguito a partire dal percorso specificato nel

parametro `Directory`; la priorità è quella normale. L'avvio del processo assegnerà un valore ai campi della variabile `proc`; la funzione `WaitForSingleObject` attenderà la variazione (**signaling**) dello stato dell'handle `hProcess` per il periodo di tempo indicato dal parametro `WaitTime`. Il valore di ritorno della funzione corrisponde a 0 in caso di signaling regolare, altrimenti `WAIT_TIMEOUT` in caso di supero del tempo massimo o altri valori in caso di errore; pertanto la funzione `ShellSospensiva` restituirà il valore di ritorno di `WaitForSingleObject` e soltanto lo 0 indicherà l'assenza di errori ed il completamento regolare.

Superata questa attesa (per signaling o per timeout) sarà necessario chiudere l'handle mediante `CloseHandle` del processo avviato (riga 50). In caso di impossibilità ad avviare il processo esterno sarà restituito il valore di ritorno corrispondente all'errore durante l'attesa `WAIT_FAILED`.

La dimostrazione del funzionamento di questa semplice funzione è quasi banale: sull'unico form sono posizionate due etichette, due caselle di testo di nome `txtProcesso` e `txtAttesa` ed un unico pulsante di nome `cmdAvvia`.



Alla pressione del pulsante `Avvia` sarà eseguito il seguente codice:

```

1. Option Explicit
2.
3. Private Sub cmdAvvia_Click()
4.     Select Case ShellSospensiva(txtProcesso.Text, CurDir$, vbNormalFocus,
5.                               CLng(txtAttesa.Text))
6.         Case 0: MsgBox "Esecuzione completata con esito positivo",
7.             vbInformation
8.         Case WAIT_TIMEOUT: MsgBox "Supero del tempo massimo di attesa",
9.             vbExclamation
10.        Case Else: MsgBox "Si è verificato un errore", vbCritical
11.    End Select
12. End Sub

```

Semplicemente chiamando la funzione `ShellSospensiva` e fornendo il percorso del processo da eseguire (`txtProcesso`), la cartella da cui avviare (in questo esempio stiamo usando la cartella corrente), lo stile della finestra (normale e attiva) ed il tempo di attesa in millisecondi (`txtAttesa`) sarà possibile recuperare il valore di ritorno per stabilire l'accaduto: il valore 0 indica l'assenza di errore e la normale apertura e chiusura del processo entro il tempo massimo; il valore corrispondente alla costante `WAIT_TIMEOUT` indica l'avvio dell'applicazione ma il superamento del tempo massimo di attesa; qualsiasi altro valore indica una situazione di errore. Per ciascuno di questi valori sarà mostrato un avviso differente:



Figura 2

Figura 3

Figura 4

L'utilizzo di questa funzione semplifica notevolmente i problemi legati alla sincronizzazione di processi esterni poiché essendo in grado di determinare l'attesa o il superamento di questo tempo è possibile effettuare le scelte opportune.

[Eibia FBI](#)

24 Aprile 2004



[Torna all'indice degli HowTo](#)
