Mandare in esecuzione un file *.inf

http://www.vbsimple.net/activity/act_08.htm

Richiesta di: Daniel - 25 Novembre 2000
Difficoltà: 1 / 5

*Devo creare un programmino EXE in Visual Basic che permetta di lanciare un file di nome **MPEG4FIX.INF**.*
*Praticamente dovrei automatizzare l'operazione che si effettua cliccando con il pulsante destro del mouse su un file *.INF e scegliendo l'opzione **INSTALLA**.*

L'operazione in questione è molto semplice.
Chiunque sappia maneggiare un po' il registro di Windows può ricavarsi la stringa che esegue Windows nel momento in cui si clicca sopra un file INF con il tasto destro del mouse e si sceglie l'opzione *Installa*.

La stringa da far eseguire al sistema è:

"RUNDLL.EXE setupx.dll,InstallHinfSection DefaultInstall 132 %1"

Naturalmente, secondo le regole del sistema Windows, al posto del %1 va inserito il nome completo del file da installare.

La soluzione che adotteremo in questo esempio è quella di inserire questo programma nella stessa cartella dove il file INF risiede.

Inseriamo sopra un form un pulsante di nome **InstallButton**. Il click sopra di esso invocherà l'installazione del file INF.

Il codice si compone di poche righe. Per comodità abbiamo inserito anche una funzione che attende la chiusura del programma di installazione del file INF. Al termine d'essa apparirà un messaggio di conferma.

```
 1. Option Explicit
 2. Private Const RigaInstallazione As String = "RUNDLL.EXE
    setupx.dll,InstallHinfSection DefaultInstall 132 "
 3. Private Const FileINF As String = "MPEG4FIX.INF"
 4.
 5. Private Sub InstallButton_Click()
 6.     Dim ReturnValue As Long
 7.     On Error Resume Next
 8.     ReturnValue = Shell(RigaInstallazione & App.Path & "\" & FileINF)
 9.     While ReturnValue <> 0
10.         AppActivate ReturnValue
11.         DoEvents
12.         If Err.Number <> 0 Then ReturnValue = 0
13.     Wend
14.     MsgBox "Installazione eseguita", vbInformation + vbOKOnly
15. End Sub
```


La riga 2 definisce una costante di nome **RigaInstallazione** che contiene la stringa da eseguire al fine di installare un file INF. Ovviamente abbiamo rimosso il %1 perché al posto di esso forniremo il nome del file INF.
Così la riga 3 definisce un'altra costante di nome **FileINF** che contiene il nome, senza percorso del file da installare, ovvero *MPEG4FIX.INF*.

All'interno della routine legata all'evento *Click* sopra il pulsante **InstallButton** abbiamo le istruzioni di installazione.
La variabile **ReturnValue** conterrà il valore ritornato dalla funzione *Shell*, valore che utilizzeremo per capire se l'installazione è terminata o meno.

In quest'operazione sfrutteremo la gestione degli errori.
Infatti non esistono funzioni per comprendere se l'esecuzione di processo esterno è terminata, senza usare l'API.
La riga 7 ci consente di non bloccare il programma nel momento in cui si verifica un errore.

La riga successiva utilizza l'istruzione *Shell* che chiama un processo esterno in maniera asincrona, ovvero non attende la chiusura del programma esterno.
Vediamo con calma questa riga:

```
ReturnValue = Shell(RigaInstallazione & App.Path & "\" & FileINF)
```

La funzione *Shell* riceve come parametro una stringa composta da 4 parti:

1. La costante **RigaInstallazione**, vista prima.
2. La proprietà **App.Path**, che ci restituisce il percorso dove si trova il programma in esecuzione, ovvero la cartella dentro la quale sono contenuti il programma ed il file INF.
3. Una piccola stringa "\" per separare il percorso dal nome del file INF.
4. La costante **FileINF** che contiene il nome, senza percorso del file INF.

L'unione dei quattro segmenti compone la corretta stringa di installazione del programma contenuto nel file ***MPEG4FIX.INF***.

La funzione *Shell* restituisce il PID (Process ID - Numero del processo) del programma chiamato.

Alla riga 9 inizia un ciclo che dura fintanto che **ReturnValue** è diverso da 0.
Se l'esecuzione del programma esterno è andata a buon fine, la variabile ReturnValue conterrà il PID del programma, che è sempre diverso da 0.

Per cui all'interno del ciclo abbiamo l'istruzione *AppActivate* che assicura che la finestra d'installazione del file INF sia in primo piano. AppActivate riceve come parametro il PID del programma da attivare.

All'interno dei cicli a lunghezza indeterminata è fondamentale inserire sempre un'istruzione *DoEvents*. Senza d'essa il nostro programma andrebbe alla massima velocità permessa dal processore, con il conseguente difetto di rallentare (se non addirittura bloccare) il programma esterno.

Nel momento in cui l'esecuzione del programma esterno chiamato attraverso la funzione *Shell* termina, non viene comunicato nulla al programma Visual Basic.
Tuttavia se continuassimo a tentare di attivare un programma ormai chiuso si verificherebbe un errore. Ma, poiché, abbiamo inserito un'istruzione *On Error*, il programma non si bloccherà all'errore detto.
All'interno di questo ciclo, alla riga 12, abbiamo un controllo degli errori verificatisi. Il verificarsi di un errore all'interno di questo ciclo identifica la fine del programma esterno. Ecco perché con la riga 12 imposteremo il valore di ReturnValue a 0. Infatti il ciclo verrà eseguito fintanto che il valore di ReturnValue è diverso da 0.
L'esecuzione dell'assegnamento ReturnValue = 0 provoca l'uscita dal ciclo.

Subito dopo il ciclo abbiamo una semplice istruzione *MsgBox* che visualizza un messaggio di conferma dell'avvenuta installazione.

Il difetto principale di questo semplice programmino è quello di non poter controllare se l'installazione del programma è andata a buon fine.
L'unica soluzione consiste nel controllare, subito dopo l'uscita del ciclo, se alcuni o tutti i files installati dal file INF sono presenti sul disco, ma questo richiede uno studio specifico del file INF in questione.

Fibia FBI
25 Novembre 2000


Torna all'introduzione delle Richieste dei lettori

