Home Page
Informazioni
Aiuto

Informazioni aggiuntive sulla compilazione condizionale

http://www.vbsimple.net/info/info_18.htm

Una caratteristica molto particolare di Visual Basic è quella di poter scrivere del codice che si comporti in maniere differenti in base ad alcuni valori costanti decisi prima della compilazione definitiva. È quindi possibile scrivere un unico progetto e compilarlo una prima volta facendo uso di certi dati ed una seconda volta utilizzando una differente soluzione.

La sua utilità si rivela in quelle rare occasioni in cui il codice sviluppato funziona correttamente con un sistema quale Windows 9x ma non funziona in ambiente Windows NT. In queste situazioni generalmente si ricorre ad una rilevazione del sistema in uso e si determina quale segmento di codice eseguire con una semplice valutazione. Tramite la compilazione condizionale invece è possibile generare due eseguibili differenti: uno studiato per il primo ambiente ed un altro studiato per il secondo ambiente.

La compilazione condizionale entra in ballo quando si vogliono includere o escludere certe parti del codice in una certa versione del programma compilato. Ad esempio è possibile aggiungere una serie di controlli, registrazioni e messaggi nella versione di sviluppo per consentire il debug quando non è possibile farlo all'interno dell'IDE di Visual Basic ed eliminare tutte le registrazioni nella versione definitiva di un software.

Ancora è possibile compilare una prima versione limitata di un software e rilasciarla per una prova gratuita. Naturalmente non c'è pirata informatico che possa resistere! Non solo il codice escluso non verrà eseguito ma addirittura non sarà nemmeno presente all'interno del file compilato.

È altresì possibile sviluppare due funzioni totalmente differenti ed incompatibili tra loro e decidere in fase di compilazione quale delle due utilizzare.

La compilazione condizionale si basa sull'uso di una istruzione base ed un costrutto di controllo:

```
#Const COSTANTE = VALORE
```

Consente di definire una costante di compilazione condizionale ed assegnarle un valore. La stessa costante può essere definita mediante le proprietà del Progetto alla voce ***Crea***. Assegnato un valore ad una costante compilazione condizionale è possibile valutare tale valore mediante il costrutto seguente:

```
#If ESPRESSIONE Then
    <istruzioni>
#ElseIf ALTRA_ESPRESSIONE Then
    <istruzioni>
#Else
    <istruzioni>
#End If
```


In base al risultato dell'espressione valutata sarà incluso o escluso il gruppo di istruzioni all'interno del controllo **#If**. Se il risultato della prima espressione restituisce il valore Falso sarà eseguito il controllo successivo **#ElseIf** oppure **#Else** (se presenti).

Passiamo ad un esempio pratico: possiamo aggiungere una serie di istruzioni di controllo per effettuare il debug del codice, che saranno eseguite soltanto quando la costante ISDEBUG è uguale a 1.

```
 1. #Const ISDEBUG = 1
 2.
 3. Private Sub Command1_Click()
 4.     Dim sngProdotto As Single
 5.     sngProdotto = Text1.Text * Text2.Text / Text3.Text
 6.     #If ISDEBUG = 1 Then
 7.         MsgBox "Text1 = " & Text1.Text
 8.         MsgBox "Text2 = " & Text2.Text
 9.         MsgBox "Text3 = " & Text3.Text
10.         MsgBox "sngProdotto = " & sngProdotto
11.     #End If
12.     Risultato = sngProdotto
13. End Sub
```

Nel momento in cui il codice verrà compilato, se il valore della costante ISDEBUG è 1 allora sarà incluso il gruppo di funzioni MsgBox all'interno del controllo #If..#End If. In caso contrario le righe 6-11 saranno escluse dalla compilazione e non appariranno in alcun modo nel file generato.
Questo comporta quindi un file eseguibile più piccolo e ripulito di istruzioni inutili nella versione definitiva.
In breve, se la costante ISDEBUG fosse stata diversa da 1, il codice risultante sarebbe stato il seguente:

```
1. Private Sub Command1_Click()
2.     Dim sngProdotto As Single
3.     sngProdotto = Text1.Text * Text2.Text / Text3.Text
4.     Risultato = sngProdotto
5. End Sub
```

Lo stesso progetto compilato con i due valori costanti produce infatti due eseguibili nettamente differenti tra loro, il cui secondo è più piccolo dell'altro.

Nell'esempio sopra citato non risalta la vera utilità della compilazione condizionale. Si provi però a pensare ad un grosso gruppo di funzioni comandate da un controllo #If. Altresì una prima versione del programma eseguibile può includere una funzione API di una certa libreria, mentre la seconda funzione può includerne un'altra di un'altra libreria. Ad esempio:

```
1. #Const WINNT = 1
2.
3. #If WINNT = 1 Then
4.     Private Declare Function ReleaseMemoryHeld Lib "MemWinNT.DLL" (ByVal lpBlock As
   Long) As Long
5. #Else
6.     Private Declare Function ReleaseMemoryHeld Lib "MemWin95.DLL" (ByVal lpBlock As
   Long) As Long
```



```
7. #End If
```

In funzione del valore della costante WINNT sarà compilato un codice piuttosto che un altro. Il codice utilizzatore della funzione potrà ad esempio non conoscere da quale libreria viene richiamata la funzione ma utilizzerà semplicemente la funzione *ReleaseMemoryHeld*.

Estendiamo ulteriormente il discorso: la versione per Windows NT ad esempio potrebbe includere una, due o centinaia di funzioni e proprietà che la versione per Windows 9x non è in grado di utilizzare. Naturalmente nella versione compilata per Windows 9x tale codice non sarà affatto presente ed il file eseguibile quindi risulterà più ridotto.

Per assurdo è addirittura possibile inserire del codice errato all'interno di una parte compilata condizionalmente. Il seguente codice ad esempio è formalmente valido:

```
1. #Const WINNT = 1
2.
3. #If WINNT = 1 Then
4.     Private hThread As Long
5. #Else
6.     Ambarabà ciccì coccò
7.     A caval donato non si guarda in bocca
8.     Questo codice non sarà neanche compilato
9. #End If
```

Il segmento di codice alle righe 6-8 non sarà neanche compilato e quindi non risulterà come errore durante l'esecuzione. All'interno del progetto compilato non vi sarà alcuna traccia di quelle istruzioni.
Naturalmente se impostassimo il valore della costante WINNT ad un valore diverso da 0 tali errori sarebbero segnalati e dall'altro lato, la variabile hThread risulterebbe inesistente.

Una reale applicazione della compilazione condizionale è presente nell'articolo Creazione di un gruppo di controlli Winsock in un modulo di classe per permettere in fase di compilazione la scelta di un array di oggetti oppure di un'istanza Collection.

Fibia FBI
3 Dicembre 2002
