

Utilizzo di un tasto per lo spostamento di campo

http://www.vbsimple.net/howto/ht_004.htm

Difficoltà:  3 / 5

Nei programmi Visual Basic, dove sono presenti vari controlli, è possibile spostare il cursore da un controllo ad un altro mediante la pressione del tasto TAB.

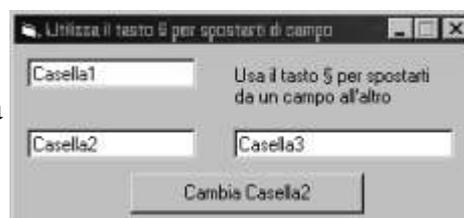
Il programma intercetta la pressione del tasto TAB e passa il [focus](#) all'oggetto la cui proprietà *TabIndex* sia successiva a quella dell'oggetto corrente. Se l'oggetto successivo è una Label **A** il focus sarà passato all'oggetto successivo in ordine di *TabIndex*.

Se un oggetto ha la proprietà *TabStop* impostata su False, non sarà possibile trasferire il focus mediante il tasto TAB.

Vedremo in questo esempio come utilizzare un altro tasto, invece del TAB, per passare da un oggetto ad un altro, da un campo all'altro. In questo esempio utilizzeremo il tasto \$ per tale operazione, poiché in molte tastiere è il tasto più vicino al tasto INVIO e ciò si rivela particolarmente utile in programmi dove è necessario immettere molti dati in varie caselle di testo come i programmi di contabilità.

Inseriamo all'interno di un form:

1. Una casella di testo di nome **Casella1** con proprietà *TabIndex* 0.
2. Una label con *TabIndex* 1.
3. Una casella di testo di nome **Casella2** con proprietà *TabIndex* 2 e proprietà *Enabled* False.
4. Una casella di testo di nome **Casella3** con *TabIndex* 3.
5. Un pulsante di nome **Command1**.



In [fase di esecuzione](#) quando l'utente preme il tasto TAB si passa dalla **Casella1** alla **Casella3** poiché non è possibile impostare il focus sopra una label o sopra un oggetto nascosto o disattivato come la **Casella2**.

Il pulsante **Command1** servirà per attivare e disattivare la **Casella2**. Scriviamo subito questo codice:

```
1. Private Sub Command1_Click()
2.     Casella2.Enabled = Not Casella2.Enabled
3. End Sub
4.
```

Al click sopra il pulsante **Command1** la proprietà booleana *Enabled* della **Casella2** sarà invertita. Passerà da True a False e viceversa.

Questo provocherà l'attivazione o la disattivazione della **Casella2**.

Passiamo subito al codice vero e proprio. È leggermente complicato come struttura, per cui

lo analizzeremo passo dopo passo.

```
5. Private Sub Form_KeyPress(KeyAscii As Integer)
```

L'evento che sfrutteremo è la pressione di un tasto ricevuta dall'oggetto Form, valido per cui in ogni momento. Altrimenti dovremmo gestirci i singoli oggetti del form.

```
6. Dim ProssimoTabIndex As Integer
```

La variabile **ProssimoTabIndex** indica il TabIndex successivo a quello del controllo attuale. Qualunque controllo con TabIndex minore a **ProssimoTabIndex** può essere scartato con sicurezza.

```
7. Dim ControlloNR As Integer
```

La variabile **ControlloNR** serve soltanto per eseguire il ciclo FOR...NEXT che controllerà tutti gli elementi del form.

```
8. Dim ControlloPrimoTab As Integer
```

Questa variabile indica qual'è il controllo adatto con il TabIndex inferiore. Questo controllo servirà nel caso che si sia raggiunta la fine dei controlli. Ad un'ulteriore pressione del tasto § il focus si dovrà spostare sul primo controllo, ovvero quello con il TabIndex inferiore.

```
9. Dim ControlloEsatto As Integer
```

La variabile **ControlloEsatto** indicherà il numero del controllo che riceverà il focus. Questo è il numero che cerchiamo all'interno di questo progetto.

```
10. Dim CercaProssimoTabIndex As Integer
11.
```

La variabile **CercaProssimoTabIndex** contiene il TabIndex più basso, ma sempre maggiore del TabIndex del controllo attuale. Ogni volta che sarà controllato un elemento verrà paragonato il suo TabIndex con la variabile **CercaProssimoTabIndex**. Se il TabIndex è superiore alla variabile, l'elemento potrà essere scartato, altrimenti sarà aggiornato il contenuto di **CercaProssimoTabIndex** con il valore di TabIndex.

```
12. If KeyAscii = Asc("§") Then
```

Innanzitutto sarà necessario controllare se il tasto premuto è il §. Se non lo è, il flusso del programma salterà alla riga 40, ovvero alla chiusura della sub.

```
13. ProssimoTabIndex = ActiveControl.TabIndex + 1
```

Viene subito memorizzato il valore del TabIndex successivo a quello del controllo attuale all'interno della variabile **ProssimoTabIndex**. Tutti i controlli con TabIndex minore di quello contenuto in tale variabile potranno essere scartati.

```
14. CercaProssimoTabIndex = -1
```

Inizializza la variabile **CercaProssimoTabIndex**.

```
15.      ControlloEsatto = -1
```

Inizializza la variabile **ControlloEsatto**.

```
16.      For ControlloNR = 0 To Controls.Count - 1
```

Qui inizia il programma vero e proprio.

Viene utilizzata la variabile **ControlloNR** per elencare tutti i controlli all'interno del form. Ricordiamo che la matrice Controls è a base 0, per cui il conteggio dovrà precedere fino a Controls.Count - 1.

```
17.      If Not TypeOf Controls(ControlloNR) Is Label Then
```

Se il controllo in analisi è una Label esso dovrà essere saltato.

Questo perché le Label, pur avendo proprietà *TabIndex*, non possono ricevere il focus.

```
18.      With Controls(ControlloNR)
```

Quest'istruzione With serve soltanto per facilitare la scrittura del codice.

Dalla riga successiva fino alla riga 34 dove sta l'End With, si potrà accedere alle proprietà del controllo in analisi - *Controls(ControlloNR)* - omettendo il nome dell'oggetto.

```
19.      If .TabStop And .Enabled And .Visible Then
```

Se il controllo ha la proprietà *TabStop* impostata a True, è attivato ed anche visibile si potrà continuare, altrimenti il controllo in esame sarà evitato.

```
20.      If .TabIndex = ProssimoTabIndex Then
```

Questo è il primo controllo degli elementi.

Se il controllo in esame ha il *TabIndex* uguale a **ProssimoTabIndex** (ovvero subito successivo a quello del controllo sopra il quale si è premuto TAB) abbiamo trovato l'oggetto della nostra ricerca.

```
21.      ControlloEsatto = ControlloNR
```

Avendo trovato il nostro controllo, aggiorniamo il contenuto della variabile

ControlloEsatto poiché essa all'uscita del ciclo FOR...NEXT dovrà contenere il numero dell'oggetto a cui donare il focus.

```
22.      Exit For
```

A questo punto è inutile continuare l'analisi.

Abbiamo già trovato il controllo che ci serviva. Possiamo forzare l'uscita dal ciclo e saltare alla riga 33.

```
23.      ElseIf .TabIndex > ProssimoTabIndex Then
```

Questa riga viene eseguita soltanto se il *TabIndex* del controllo in analisi è diverso da **ProssimoTabIndex**. Se il controllo in esame ha un *TabIndex* inferiore a quello della variabile potrà essere saltato tranquillamente, poiché stiamo eseguendo un passaggio di campo in avanti e non all'indietro, per cui il *TabIndex*, per essere ritenuto adatto dovrà essere maggiore di **ProssimoTabIndex**.

```
24.           If .TabIndex < CercaProssimoTabIndex Then
```

La variabile **CercaProssimoTabIndex** conterrà il **TabIndex** più basso trovato durante l'esecuzione del ciclo **FOR...NEXT**. Si andrà per tentativi. Si vaglieranno tutti i controlli fino a trovare quello con il **TabIndex** più basso, ma consecutivo a **ProssimoTabIndex**. Se si dovesse trovare un controllo con **TabIndex** inferiore a quello contenuto nella variabile **CercaProssimoTabIndex**, abbiamo trovato un possibile candidato ad essere il controllo che riceverà il focus.

```
25.           CercaProssimoTabIndex = .TabIndex
```

Trovato questo candidato, cambiamo il contenuto della variabile **CercaProssimoTabIndex** con il valore della proprietà **TabIndex** del controllo in esame.

Nei cicli successivi, se dovesse esserci un controllo con **TabIndex** inferiore, verrà diminuito questo valore, così alla fine di tutti i cicli, il valore contenuto nella variabile **CercaProssimoTabIndex** sarà con sicurezza il **TabIndex** del controllo che riceverà il focus.

```
26.           ControlloEsatto = ControlloNR
```

Man mano che **CercaProssimoTabIndex** diminuisce di valore, viene aggiornata anche la variabile **ControlloEsatto**. Essa servirà per indicare il numero del controllo con il **TabIndex** più basso.

```
27.           ElseIf CercaProssimoTabIndex < 0 Then
```

Un caso a parte è la prima esecuzione del ciclo **FOR...NEXT**. Infatti prima di iniziare tale ciclo, la variabile **CercaProssimoTabIndex** viene inizializzata a -1. In questa strana iniziale situazione, qualunque sia il controllo in esame, viene reputato come valido candidato per ricevere il focus.

Sarà la riga 24 che controllerà se gli altri controlli avranno un **TabIndex** inferiore a quello analizzato per primo.

```
28.           CercaProssimoTabIndex = .TabIndex
```

Ed ecco che il primo controllo (ovvero quando la variabile **CercaProssimoTabIndex** è inferiore a 0) viene ritenuto valido per ricevere il focus.

```
29.           ControlloEsatto = ControlloNR
```

E con questa riga viene pure memorizzato il numero del controllo, che per primo, potrebbe essere il controllo che riceverà il focus.

```
30.           End If
```

Questa riga chiude i due controlli: quello della riga 24, con **TabIndex** inferiore a **CercaProssimoTabIndex** e quello del primo ciclo, ovvero con **CercaProssimoTabIndex** inferiore a 0 (riga 27).

```
31.           End If
```

Questa chiusura si riferisce all'uscita del controllo effettuato alla riga 20 o alla riga 23.

```
32.           If .TabIndex < Controls(ControlloPrimoTab).TabIndex Then
    ControlloPrimoTab = ControlloNR
```

Con questa riga, eseguita ogni volta, memorizziamo nella variabile **ControlloPrimoTab** il numero del controllo con il `TabIndex` più basso in assoluto, anche più basso di quello su cui è stato premuto il tasto TAB.

Questo valore ci servirà nel caso ci trovassimo sull'ultimo controllo ed un'ulteriore pressione del tasto TAB richiederebbe che il focus passasse al primo oggetto attivo.

```
33.           End If
```

Questa riga chiude il controllo effettuato alla riga 19, ovvero quello della validità del controllo in esame. Un controllo disabilitato, nascosto, o con `TabStop` impostato a `False` non viene neanche analizzato.

```
34.           End With
```

Da qui in poi non verrà utilizzato il controllo preso in esame come oggetto predefinito nei riferimenti. Questa riga chiude quella aperta alla riga 18.

```
35.           End If
```

Questa riga chiude la riga 17 di controllo del tipo di oggetto. Se l'oggetto in esame fosse stato una `Label`, il flusso sarebbe passato direttamente in questa riga, evitando tutti i controlli visti in precedenza.

Ricordiamo che le `Label` non possono ricevere il focus.

```
36.           Next ControlloNR
```

Qui si chiude il ciclo `FOR...NEXT`.

Questa riga aggiorna il contenuto di **ControlloNR** e riporta il flusso delle istruzioni alla riga 16, fino a che **ControlloNR** non contenga un valore maggiore o uguale al numero dei controlli presenti nel form.

```
37.           If ControlloEsatto < 0 Then ControlloEsatto = ControlloPrimoTab
```

Questa riga controlla se la variabile **ControlloEsatto** contiene ancora il valore -1 dato nell'inizializzazione.

La presenza di tale valore indica che non si è trovato nessun candidato a ricevere il focus, quindi ci troviamo sull'ultimo controllo del form.

Quest'istruzione imposterà **ControlloEsatto** uguale al controllo con il `TabIndex` più basso, indicato da **ControlloPrimoTab** e impostato alla riga 32.

Qui abbiamo finito tutti i controlli del caso. Adesso abbiamo con sicurezza trovato il numero del controllo con il `TabIndex` successivo a quello del controllo corrente, oppure abbiamo il numero del controllo con `TabIndex` più basso.

In ogni caso la variabile **ControlloEsatto** conterrà il numero ordinale del controllo che

dovrà ricevere il focus.

```
38.     Controls(ControlloEsatto).SetFocus
```

Ed ecco finalmente la promozione!

Il controllo con numero uguale a quello contenuto in **ControlloEsatto** può ricevere il focus, mediante l'istruzione *SetFocus*.

```
39.     KeyAscii = 0
```

Prima di uscire dalla funzione ricordiamoci di azzerare il tasto premuto, altrimenti se il controllo attivo fosse una casella di testo, essa riceverebbe il carattere § e lo scriverebbe nel suo testo.

Quest'ultima istruzione assicura che il tasto premuto non venga scritto.

```
40.     End If
```

Qui si chiude il primo controllo fatto all'inizio della funzione (riga 12), ovvero il controllo del tasto premuto.

```
41. End Sub
```

E qui finalmente si chiude la nostra Sub. All'uscita d'essa sicuramente il controllo attivo sarà quello successivo a quello sopra il quale è stato premuto il tasto TAB.

Abbiamo adottato questa soluzione perché, anche se un po' complessa, comprende tutti i controlli del caso.

Avremmo potuto adottare soluzioni più semplici ma non avremmo avuto la sicurezza che abbiamo con questa. In questo modo il programmatore può in ogni momento attivare, disattivare, nascondere, creare e distruggere oggetti sopra il form. La funzione appena vista, infatti, analizza **ogni volta tutti** i controlli presenti sul form e sceglie quello con il TabIndex successivo, oppure ritorna al primo dei controlli.

[Giuseppe Della Bianca](#)
27 Novembre 2000



[Torna all'indice degli HowTo](#)
