



Proprietà Enabled dei controlli ScrollBar

http://www.vbsimple.net/weird/wrd_04.htm

Difficoltà: ► 1 / 5

Quasi tutti i controlli [Thunder](#) possiedono una [proprietà](#) **Enabled** che determina se il controllo è in grado di ricevere informazioni dall'utente. Quando il suo valore è impostato su *False* tutte le azioni dell'utente sono ignorate ed il controllo non riceve alcuna notifica degli eventi.

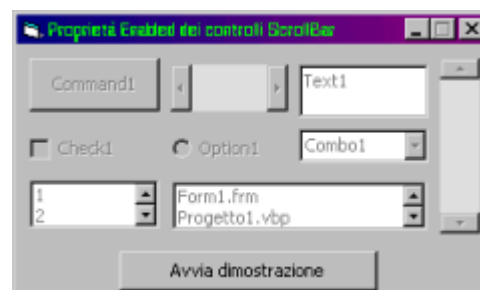
Questo tuttavia non avviene quando gli eventi sono generati - direttamente o indirettamente - dal codice del progetto. Ad esempio se il codice va a modificare il contenuto di una casella di testo la cui proprietà *Enabled* è impostata su *False*, sarà comunque generato l'evento *Change* della casella stessa.

In alcune situazioni è necessario far modificare al codice il valore di certi controlli, ad esempio svuotare le caselle di testo senza tuttavia per questo eseguire il codice associato all'evento *Change* che potrebbe avere effetti collaterali. Una soluzione al problema potrebbe essere quella di testare il valore della proprietà *Enabled* all'interno di quelle procedure critiche e **non eseguire** le azioni associate all'evento se il valore della proprietà è *False*.

Mentre questa soluzione è ragionevolmente funzionale per i controlli più comuni, così non è per i controlli di scorrimento **HScrollBar** e **VScrollBar** poiché essi si comportano in una maniera del tutto differente e fuorviante dalla normale logica.

In un progetto reale questo comportamento anomalo mi ha fatto perdere almeno un'ora di [debug](#) su un codice apparentemente perfetto. Per riprodurre la situazione sarà sviluppato un progetto davvero molto semplice che conterrà tutti i controlli Thunder visibili e che possiedono la proprietà *Enabled*.

Escludendo il pulsante in basso, tutti i controlli sono piazzati a casaccio mantenendo i loro nomi e valori originali (Text1, Combo1 ...) ed impostando la loro proprietà *Enabled* a *False*. Giusto per semplificare, il controllo *ListBox* sarà riempito a [runtime](#) con i numeri da 1 a 10.



Il pulsante in basso con il testo **"Avvia dimostrazione"** prenderà il nome **cmdAvvia** ed a differenza di tutti gli altri controlli, questo non sarà disabilitato.

È necessario quindi aggiungere una semplice riga di codice per ogni controllo da verificare. Il codice leggerà semplicemente il valore della proprietà **Enabled** e ne mostrerà il valore con una finestra di messaggio. Questa singola riga di codice andrà inserita

nell'evento *Change* se esiste oppure nell'evento *Click*, in mancanza dell'altro evento.

Il codice sarà come il seguente:

```
1. Private Sub Combol_Change()  
2.     MsgBox "Combol.Enabled = " & Combol.Enabled  
3. End Sub  
4.
```

Naturalmente ogni controllo avrà il nome del suo controllo. Così Command1_Click apparirà come:

```
5. Private Sub Command1_Click()  
6.     MsgBox "Command1.Enabled = " & Command1.Enabled  
7. End Sub  
8.
```

Potrà sembrare noiosa come operazione ma è necessaria per distinguere i singoli controlli durante la dimostrazione. Alternativamente si potrebbe costruire una semplice routine che ricevendo un argomento di tipo Control, mostri la finestra di avviso leggendo il nome del controllo ed il valore della proprietà Enabled.

Passiamo al codice vero e proprio, quello del pulsante **cmdAvvia**:

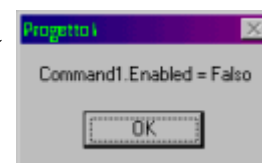
```
42. Private Sub cmdAvvia_Click()  
43.     Command1.Value = True  
44.     HScroll1.Value = Rnd * HScroll1.Max  
45.     Text1.Text = String$(10, Rnd * 25 + 65)  
46.     Check1.Value = Abs(Check1.Value - 1)  
47.     Option1.Value = False  
48.     Option1.Value = True  
49.     Combol.Text = Text1.Text  
50.     List1.ListIndex = Rnd * List1.ListCount  
51.     File1.ListIndex = Rnd * File1.ListCount  
52.     VScroll1.Value = Rnd * VScroll1.Max  
53. End Sub
```

Fondamentalmente molto banale, richiama uno per uno gli eventi *Change* o *Click* dei controlli inseriti nel form. La riga 43 ad esempio attiva l'evento *Click* del pulsante **Command1** (in una maniera vista in [un HowTo](#)), il quale lancerà la routine Command1_Click(), mostrando quindi il valore della proprietà Enabled all'interno della routine.

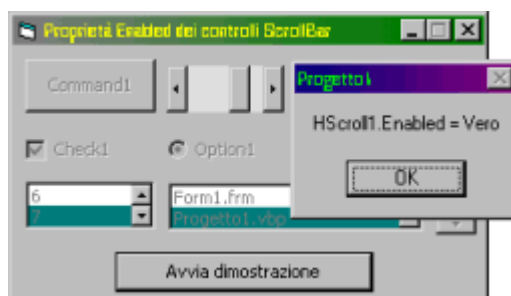
La riga 44 invece sceglierà un valore casuale nell'intervallo compreso tra 0 ed il valore massimo della barra di scorrimento **HScroll1**. In maniera analoga anche la riga 52 per il controllo **VScroll1**.

Tutte le righe sono atte a modificare il valore corrente di ogni controllo.

L'esecuzione del progetto mostrerà innanzitutto il valore della proprietà Enabled del controllo Command1 all'interno della routine Command1_Click. Altrettanto faranno le altre routine, ciascuna per il suo controllo.



Il comportamento anomalo si verificherà soltanto per i controlli **HScrollBar** e **VScrollBar**. Sebbene essi siano disattivati, il test nella routine HScroll1_Change e VScroll1_Change mostrerà il valore **True** della proprietà Enabled (vedi Figura 3 in basso).

**Figura 3**

Un'analisi più approfondita mostrerà che in realtà la barra di scorrimento viene attivata solo un attimo, nel momento in cui scatta l'evento Change e si disattiva automaticamente all'uscita dell'evento.

Possiamo notare infatti la differenza di colorazione tra la barra disattivata inizialmente (Figura 4) e la barra attivata durante l'esecuzione dell'evento Change (Figura 5). Possiamo anche notare nella Figura 5 la barretta che indica il valore corrente della barra di scorrimento.

**Figura 4****Figura 5**

Questo strano comportamento rende inutile la verifica del valore della proprietà Enabled all'interno degli eventi Change delle barre di scorrimento. Se si rende necessario inibire temporaneamente l'azione dell'evento Change sarà quindi necessario trovare una soluzione alternativa, utilizzando variabili Statiche o esterne all'evento.

Per tutti gli altri controlli Thunder, la proprietà Enabled si comporta nella maniera corretta ed il controllo non si attiva e disattiva automaticamente.

[Fibia FBI](#)

29 Settembre 2002

[Torna all'indice delle Stranezze](#)