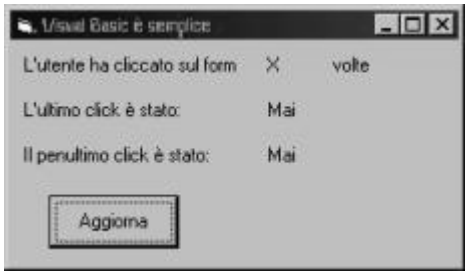
Corso base - Lezione 7

http://www.vbsimple.net/base/bas_07.htm

Cercheremo in questa lezione di dare al nostro progetto la forma di un vero programma, vedremo infatti i primi pulsanti di Visual Basic. Per cui riapriamo il nostro Progetto1 e diamogli una impostazione migliore. Invece di aggiornare i dati quando l'utente clicca sopra la superficie del form, inseriremo un vero pulsante al cui click si aggiorneranno i dati.

Osserviamo la casella degli strumenti e troveremo facilmente l'icona del controllo ***CommandButton*** , il pulsante di comando, utilizzabile per effettuare le operazioni più disparate. Andiamo quindi nella finestra di progettazione del form, ormai lo dovreste saper fare senza ulteriori istruzioni, e qui inseriamo un pulsante di questo genere. La posizione non è importante.

La prima cosa che noteremo appena inserito il pulsante è che sopra d'esso c'è una scritta: **Command1**. Ebbene, come i form e le Label, anche la classe CommandButton ha la proprietà *Caption*, cioè la proprietà che definisce il testo visibile all'utente. Come prima cosa diamo una caption migliore al nostro pulsante scrivendoci dentro ***Aggiorna***, per renderlo più comprensibile. Cambiamo anche il nome del pulsante per abituarci a scrivere codici più semplici ed ordinati. Per cambiare il nome a qualsiasi oggetto, selezionare l'oggetto cliccandoci sopra e modificare il valore della proprietà *Name* di tale oggetto. Diamogli il nome di ***PulsanteConta***, naturalmente tutta una parola, poiché tutti i nomi degli oggetti devono essere formati da una sola parola.
Al termine di questo il nostro form dovrebbe apparire simile alla figura mostrata a lato. Tuttavia se proviamo il programma e clicchiamo il pulsante non succede nulla, perché non è stato ancora impostato l'evento Click del pulsante. Per cui andiamo nella finestra del codice, selezioniamo dalla lista degli oggetti, l'oggetto PulsanteConta e selezioniamo sulla destra l'evento Click. Verrà generata in automatico la routine PulsanteConta_Click.



Ora *trasferiamo* le istruzioni dalla routine Form_Click alla nuova routine.

La maniera più semplice è sfruttare le funzionalità di *Copia/Incolla* di Windows, ovvero selezionare con il mouse o con la tastiera la parte di codice che ci interessa (nel nostro caso tutto il contenuto della routine Form_Click, escluse la riga `Private Sub Form_Click()` e la riga `End Sub`. Quelle infatti non fanno parte della routine, esse sono soltanto l'intestazione e

la chiusura della routine. Selezionato il nostro codice, apriamo il menu Modifica e clicchiamo sulla voce ***Copia***. Fatto questo posizioniamo il cursore del testo all'interno della routine PulsanteConta_Click, cioé tra la sua intestazione e la sua fine, riapriamo il menu Modifica e clicchiamo la voce Incolla. Automaticamente verrà copiato il testo precedentemente selezionato. A questo punto è possibile cancellare l'intera routine Form_Click, dalla sua definizione alla sua fine.

Le funzioni di *Copia/Incolla* non sono funzionalità di Visual Basic, ma sono presenti in quasi tutti i programmi per Windows, ragion per cui non verranno approfondite in questa sede.

Se avete fatto tutto correttamente dovreste avere il seguente codice:

```
 1. Private Const PIGreco As Double = 3.14159265358979
 2.
 3. Private Type ClickConData
 4.     NumeroClick As Integer
 5.     UltimoClick As Date
 6.     PenultimoClick As Date
 7. End Type
 8.
 9. Dim MioClick As ClickConData
10.
11. Private Sub PulsanteConta_Click()
12.     MioClick.NumeroClick = MioClick.NumeroClick + 1
13.     MioClick.PenultimoClick = MioClick.UltimoClick
14.     MioClick.UltimoClick = Now
15.
16.     Label2.Caption = MioClick.NumeroClick
17.     Label5.Caption = MioClick.UltimoClick
18.     Label7.Caption = MioClick.PenultimoClick
19.
20.     Me.Caption = PIGreco
21. End Sub
```

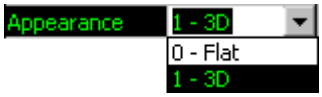
Proviamo ora ad eseguire il programma e vediamone i risultati.
Adesso, com'è prevedibile, cliccando sopra il form non accade e non cambia nulla, ma se clicchiamo sul pulsante **Aggiorna** viene eseguito il conteggio dei click.

Torniamo alla finestra di progettazione del form e selezioniamo l'oggetto PulsanteConta e proviamo ad osservarne le proprietà nella finestra delle proprietà.

La prima che troviamo è la proprietà ***Name***, la cui utilità l'abbiamo già vista, ma soffermiamoci un minuto. In questo momento il pulsante si chiama PulsanteConta ed abbiamo una routine di nome PulsanteConta_Click. Se ora decidessimo di cambiare il nome al pulsante, tutto quello che si trova all'interno della finestra del codice non verrà toccato. Per cui la routine legata al click del pulsante non funzionerà più. Dovremmo rifarla da capo o aggiustare i riferimenti a tali oggetti.

La scelta migliore, da adottare sempre ed in ogni caso, è quella di decidere i nomi degli oggetti prima di cominciare a scrivere codice legato all'oggetto, direttamente (come le routine di eventi) e indirettamente (come le istruzioni che fanno riferimento all'oggetto).

La seconda proprietà è ***Appearance***, e come leggiamo nella finestrella sotto le proprietà serve ad indicare al programma se l'oggetto verrà ridisegnato con effetti tridimensionali.



Osserviamo che la proprietà ha una freccetta accanto al valore, se clicchiamo sulla freccetta ci verranno mostrati i possibili valori della proprietà. Questa proprietà ha solo due valori ammissibili. Non potremo scrivere dentro la proprietà un valore come abbiamo fatto per le proprietà Caption e Name, ma saremo costretti a sceglierlo dalla lista.

Ormai quasi tutti i programmi per Windows hanno le finestre in 3D, con i bordi rialzati o abbassati. Impostare questa proprietà a 0 - Flat solo se vogliamo dare al nostro programma un look diverso da tutti gli altri programmi. Noi lo lasceremo a 1 - 3D.

Segue nell'elenco delle proprietà ***BackColor***, che definisce il colore che avrà il pulsante. Anche questa proprietà ha la freccetta di scelta del valore. Clicchiamo su d'essa e si aprirà una finestrella per la scelta del colore. Questa si divide in due zone: Tavolozza e Sistema. La prima serve per scegliere un colore arbitrariamente, mentre la seconda fornisce i colori di sistema impostati per tutti i programmi standard Windows. Gran parte degli utenti utilizzano il grigio come sfondo delle finestre, ma attraverso il *pannello di controllo di Windows* è possibile decidere il colore standard delle finestre. Se noi impostassimo un colore attraverso la Tavolozza, l'utente che ha un diverso colore per le finestre continuerebbe a vedere il controllo con il colore impostato da noi. La scelta che noi adotteremo è quella di lasciare i colori predefiniti, in questo caso il colore grigio di sistema.

Segue la proprietà ***Cancel***, che stabilisce se il pulsante è un pulsante Annulla della finestra. Impostare questo valore a True fa sì che quando l'utente preme il tasto ESC sulla finestra viene chiamato l'evento Click del pulsante, anche se l'utente non ha esplicitamente cliccato con il mouse sopra d'esso. L'effetto di annullamento descritto all'interno di VB deve essere impostato dal programmatore.

La proprietà successiva è ***Caption***,che abbiamo già visto. Essa serve ad impostare il testo sopra il pulsante.

La proprietà ***Default*** fa sì che quando l'oggetto attivo sia un controllo che non gestisce il tasto ENTER, e l'utente, in quella situazione, preme detto tasto, viene eseguito l'evento Click del tasto.

La proprietà ***DisabledPicture*** serve ad impostare un'immagine che verrà mostrata quando il pulsante viene disabilitato attraverso la proprietà Enabled.

La proprietà ***DownPicture*** imposta l'immagine che sarà mostrata nel momento in cui l'utente tiene premuto il tasto del mouse, per esempio quando clicca, sopra il controllo.

La proprietà ***DragIcon*** imposta l'icona che verrà visualizzata in caso di trascinamento. Il trascinamento sarà trattato nel corso Avanzato, per cui ignorare questa proprietà per il momento.

La proprietà ***DragMode*** definisce la modalità di trascinamento, automatica o manuale.

La proprietà ***Enabled*** attiva e disattiva un controllo. Quando essa è impostata a False il controllo non genererà nessun evento quando l'utente cerca di operare con essa. In genere si utilizza questa proprietà quando si vuole bloccare l'utilizzo di un pulsante finché non si verifica un determinato evento.

La proprietà ***Font*** imposta il carattere utilizzato nelle operazioni di scrittura sul controllo, come la proprietà Caption. Questa è una macroproprietà, nel senso che racchiude al suo interno tante proprietà. Infatti il programmatore via codice potrebbe impostare questa proprietà complessa o potrebbe modificare le singole componenti: *FontName, FontSize, FontBold, etc.*.

La proprietà ***Height*** definisce l'altezza dell'oggetto.

HelpContextID serve per impostare la pagina della guida collegata a quel pulsante. La generazione di una guida di un programma sarà trattata in un altro corso.

La proprietà ***Index*** è l'indice del controllo in una matrice di controlli. Le matrici saranno trattate nel corso Intermedio.

La proprietà ***Left*** definisce la posizione dell'oggetto rispetto al bordo sinistro del suo contenitore.

La proprietà ***MaskColor*** serve a definire una maschera utilizzata per creare un effetto di trasparenza.

Le proprietà ***MouseIcon*** e ***MousePointer*** definiscono la forma del puntatore del mouse. La seconda permette di scegliere un puntatore tra quelli impostati nel sistema oppure uno personalizzato (voce *Custom*). Quando la proprietà MousePointer è impostata a Custom verrà utilizzato come puntatore quello specificato attraverso la proprietà MouseIcon.

La proprietà ***OLEDropMode*** interviene in operazioni di trascinamento, trattate in un altro corso.

La proprietà ***Picture*** serve per inserire un'immagine che verrà visualizzata sul pulsante quando la proprietà Style è impostata su Graphical.

La proprietà ***RightToLeft*** serve a definire la direzione di scrittura del testo. Non è disponibile nelle lingue occidentali, ma soltanto nelle edizioni di Visual Basic in lingue che permettono la scrittura da destra verso sinistra.

Come accennato prima la proprietà ***Style*** definisce lo stile del pulsante. Sia esso normale o grafico, cioè con un'immagine sopra. Se il valore è True verrà visualizzata l'immagine impostata nella proprietà Picture.

La proprietà ***TabIndex*** stabilisce l'ordine di tabulazione nel form. Questa proprietà determina l'ordine con il quale l'oggette riceve il Focus sfruttando il tasto TAB o le freccette.

La proprietà ***TabStop*** determina se l'oggetto può ricevere il focus utilizzando il tasto TAB.

La proprietà ***Tag*** non viene utilizzata mai ed è disponibile in tutti i controlli per memorizzare qualche dato utile. Il suo valore non determina il cambiamento di nessuno stato.

La proprietà ***ToolTipText*** imposta il testo che apparirà quando l'utente posiziona il mouse sopra il pulsante per un secondo circa.

La proprietà ***Top*** determina la posizione verticale dell'oggetto dal bordo superiore del suo contenitore.

La proprietà ***UseMaskColor*** determina se il colore impostato nella proprietà MaskColor verrà utilizzato per creare aree trasparenti.

La proprietà ***Visible*** imposta la visibilità dell'oggetto. Un oggetto nascosto funziona a tutti gli effetti, ma non risponde agli eventi del mouse.

La proprietà ***WhatsThisHelpID*** identifica un argomento della guida da utilizzare quando è presente un pulsante *What's This*. La generazione di un file della guida sarà trattata in un altro corso.

L'ultima proprietà della finestra delle proprietà è la ***Width*** che determina la larghezza del pulsante.

Queste sono tutte le proprietà disponibili in fase di progettazione per l'oggetto CommandButton. Ma ci sono anche alcune proprietà utilizzabili solo in fase di esecuzione; queste sono:

Container un riferimento all'oggetto dentro il quale il pulsante è contenuto.

hWnd, handle del pulsante, utilizzato dall'API per effettuare operazioni su di esso.

Parent, simile a Container, che identifica il Form dentro il quale risiede il pulsante, e non il contenitore.

Value che identifica lo stato del pulsante, premuto o non premuto.

Di tutte queste numerose proprietà inizialmente utilizzeremo soltanto *Name, Caption, Height, Left, Top, Visible e Width*.

Passiamo ora agli eventi dell'oggetto CommandButton. Abbiamo:

Click, evento principale. Determina il momento in cui l'utente clicca con il mouse sopra il controllo.

DragDrop DragOver, OLECompleteDrag, OLEDragDrop, OLEDragOver, OLEGiveFeedBack, OLESetData e ***OLEStartDrag*** collegati alle operazioni di trascinamento. Saranno visti in un altro corso.

GotFocus, eseguito nel momento in cui il pulsante diventa l'oggetto attivo.

KeyDown, KeyUp e ***KeyPress*** eseguiti rispettivamente nei momenti quando l'utente preme un tasto, quando lo rilascia e quando viene inviato un carattere al pulsante.

LostFocus, eseguito nel momento in cui il pulsante cessa di essere l'oggetto attivo.

MouseDown e ***MouseUp***, generati nel momento in cui l'utente preme il pulsante del mouse e nel momento in cui lo lascia. Subito dopo l'esecuzione dell'evento MouseUp viene generato l'evento Click.

MouseMove, eseguito ogni volta che il puntatore del mouse si sposta sopra la superficie del pulsante.

Quello che abbiamo visto è uno dei controlli più semplici. Tuttavia possiede tantissime proprietà e vari eventi. Piano piano impareremo ad usare quelli che ci servono. Intanto non facciamoci impressionare dalla loro quantità. Per i normali programmi ci basteranno poche proprietà e ancora meno eventi.

Fibia FBI
30 Ottobre 2000

Torna alla sesta lezione Vai all'ottava lezione

