Classi, oggetti ed eventi personalizzati

In VBA è possibile creare oggetti personalizzati attraverso la definizione di classi e l'inserimento di moduli di classe. È inoltre possibile creare eventi di classe e procedure di evento personalizzati integrati in Excel. Con questa premessa risulta abbastanza chiaro che tramite VBA è possibile definire e creare delle classi di oggetti che possano soddisfare le esigenze del singolo utente. Per definizione la programmazione orientata agli oggetti prevede di raggruppare in un'unica entità, ovvero in una classe, sia la struttura dati che le procedure che operano su di essa, creando un oggetto unico dotato di proprietà e metodi che operano sull'oggetto stesso.

La classe è un modello per i nuovi oggetti che verranno creati e sono utilizzate per memorizzare, elaborare e rendere disponibili i dati, in quanto oltre ai dati contengono anche il codice per gestirli con procedure e metodi come le *Sub* e le *Function*. Una classe viene creata inserendo un modulo di classe nel progetto VBA e consente di creare i propri oggetti con proprietà e metodi molto simili ad altri oggetti come Range, foglio, grafico, etc. Il modulo di classe ha una serie di procedure che includono variabili e costanti che definiscono le sue proprietà che possono essere manipolate in un modulo di classe con le procedure *Property Let*, *Property Get* e *Property Set*. Per accedere alle proprietà e metodi dell'oggetto classe da una routine in un modulo di codice, si dichiara una variabile oggetto del tipo della classe.

Si può programmare in VBA, senza creare oggetti personalizzati che in realtà non aumentano la funzionalità del codice, tuttavia, l'utilizzo di oggetti personalizzati rende la codifica meno complessa e più semplice avendo un collegamento nel codice rendendo la codifica auto documentata denominando le classi in modo appropriato, e questo aiuta il debug e il codice riutilizzo.

Inserire un modulo di classe

In Visual Basic Editor (VBE), per inserire un modulo di classe si deve cliccare su **Inserisci - Modulo di classe**, in alternativa, nella finestra dei *progetti*, cliccare con il pulsante destro del mouse sul nome del progetto e sul menu che appare cliccare su *Inserisci* e quindi scegliere *Modulo di classe*. In alternativa, sulla barra degli strumenti standard in VBE, cliccare sul pulsante *Inserisci* e quindi scegliere *Modulo di classe*. Questo crea una classe vuota con il nome di *Class1*. Per rimuovere o eliminare un modulo di classe, si deve cliccare con il tasto destro del mouse dopo aver selezionato il modulo nella finestra dei progetti e quindi fare clic su *Rimuovi*

Le istanze di proprietà di un modulo di classe

C'è una grande differenza tra una variabile semplice e una variabile oggetto, la variabile oggetto non è che un puntatore all'interno della memoria e si dovrà creare esplicitamente un oggetto e salvare la sua posizione nella variabile oggetto. Questo processo si chiama creare una nuova istanza di un oggetto o*istanziare un oggetto*.
Poiché gli oggetti sono diversi dalle variabili, Visual Basic for Applications utilizza una speciale istruzione chiamata istruzione *Set* che ha due forme.

1) *Set VariabileOggetto = New NomeClasse*

In questa forma, l'istruzione Set crea un nuovo oggetto basato su *NomeClasse*, ciò significa che Visual Basic allocherà memoria per l'oggetto e salverà la posizione in memoria nella classe *VariabileOggetto*.

2) *Set VariabileOggetto = EspressioneOggetto*

Nella seconda forma, l'istruzione *Set* fa due cose: prima di tutto rilascia l'oggetto a cui puntava, quindi salva un puntatore a un oggetto già esistente in *VariabileOggetto*.

La proprietà *Instancing* di un modulo di classe è impostata su *Private* di default che non permette a un progetto esterno di lavorare con le istanze di quella classe, si deve impostare la proprietà *Instancing* per *PublicNotCreatable* per consentire a progetti esterni, con un riferimento al progetto contenente la classe definita, per accedere e utilizzare istanze della classe. Notare che l'impostazione di PublicNotCreatable ancora non consente al progetto esterno di istanziare (cioè creare o chiama all'esistenza) l'oggetto classe o un'istanza della classe, che può essere istanziato solo dal progetto che contiene la definizione della classe. Si noti che il progetto esterno può utilizzare un'istanza della classe definita se il progetto di riferimento ha già creato tale istanza.

Come già accennato, in VBA è possibile creare i propri oggetti personalizzati attraverso la definizione di classi. Una classe viene creata inserendo un modulo di classe e per accedere alle proprietà e metodi della classe dell'oggetto da una routine in un modulo di codice, è necessario creare una nuova istanza dell'oggetto di classe. Si noti che possono essere create pluralità di istanze a un oggetto di classe. Ci sono due modi per creare un'istanza, uno con un codice a due-line o in alternativa con un codice a riga singola.

Per creare un'istanza di una classe Two-line si deve utilizzare l'istruzione Dim per creare una variabile (studente) e definirlo come un riferimento alla classe (Studenti): *Dim studente As Studenti*, mentre per creare un nuovo riferimento oggetto utilizzando la parola chiave *New*, indicando il nome della classe (Studenti) che si desidera creare un'istanza, dopo la parola chiave New: *Set studente = New Studenti*

In alternativa si può usare un codice a linea singola per creare un'istanza di una classe, in questo caso l'oggetto *Studenti* viene istanziato solo quando il metodo di classe viene chiamato *Dim studente As New Studenti*

Creare Proprietà della classe

Un modo per creare una proprietà di classe è quello di dichiarare una variabile pubblica nel modulo di classe, e questa proprietà sarà in lettura-scrittura, l'altro modo è quello di utilizzare procedure di proprietà per creare una variabile privata per contenere i valori e utilizzare istruzioni di proprietà (cioè Property Let, Property Set e Property Get). La creazione di proprietà mediante una variabile pubblica, anche se semplice, non è generalmente preferibile perché non è flessibile, mentre invece utilizzando le dichiarazioni di proprietà sarà possibile impostare una proprietà di sola lettura o sola scrittura in aggiunta a lettura-scrittura, mentre l'uso di una variabile pubblica creerà solo le proprietà in lettura-scrittura. Inoltre, utilizzando le istruzioni della struttura è possibile eseguire il codice per calcolare i valori come proprietà che utilizza, mentre una variabile pubblica non consente l'uso di codice per impostare o restituire il valore di una proprietà.

Creare metodi in un modulo di classe

Oltre alle proprietà, gli oggetti possono anche avere uno o più metodi, un metodo è definito come *Sub* e *Funzioni* ed è creato con le procedure di sub-routine e funzioni. Un metodo è una sub-routine contenente un insieme di codici che eseguono un'azione o un'operazione sui dati all'interno della classe, o una funzione che contiene un insieme di codici che restituisce un valore dopo l'esecuzione di un'operazione. In un modulo di classe, solo se il metodo è dichiarato pubblico può essere chiamato da un'istanza di questa classe, altrimenti se un metodo viene dichiarato privato può essere chiamato solo da altri metodi all'interno della classe. Si noti, che di default la procedura è pubblica se non vengono specificate le parole chiave pubbliche o private.

Utilizzo di procedure di proprietà

Le procedura di proprietà sono un insieme di codici che creano e manipolano le proprietà personalizzandole per un modulo di classe. Una procedura *Property*è dichiarata da una dichiarazione *Property Set*, *Property Let* o *Property Get* e termina con un'istruzione *End Property*. *Property Let* viene utilizzata per assegnare un valore di *sola scrittura* a una proprietà e *Property Get* restituisce o recupera il valore di una proprietà di *sola lettura*, che può essere solo restituito, ma non impostato. *Property Set* assegna un valore di *sola scrittura* e viene utilizzata per impostare un riferimento a un oggetto. Le procedure di proprietà sono solitamente definite in coppia, *Property Let* e *Property Get* o *Property Set* e *Property Get*, si tenga presente che una procedura creata con*Property Let* consente all'utente di modificare o impostare il valore di una proprietà, mentre l'utente non può impostare o modificare il valore di una proprietà di sola lettura (cioè Property Get).

Una procedura di proprietà può fare tutto ciò che è consentito all'interno di una routine, come eseguire un'azione o un calcolo sui dati. Una procedura Property Let (o Property Set) è una procedura indipendente che può passare argomenti, eseguire azioni come da un insieme di codici e cambiare il valore dei suoi argomenti, come una routine Get o una funzione, ma non restituisce un valore come loro. Una procedura per ottenere la proprietà è anche una procedura indipendente che possa passare argomenti, eseguire azioni come da un insieme di codici e cambiare il valore dei suoi argomenti, come una procedura Property Let (o Property Set), e può essere usata in modo simile a una funzione che ritorni il valore di una proprietà.

Una procedura Property Get accetta un argomento in meno della dichiarazione Property Let o Property Set e deve essere dello stesso tipo di dati come il tipo di dati dell'ultimo argomento nella dichiarazione Property Let o Property Set associata. La dichiarazione Property Get utilizzerà lo stesso nome di proprietà come utilizzato nella dichiarazione Property Let o Property Set associata.

Una procedura Property Let può accettare più argomenti, e in questo caso l'ultimo argomento contiene il valore da assegnare alla proprietà, questo ultimo argomento nella lista degli argomenti è il valore della proprietà impostata dalla procedura chiamante. Il nome e il tipo di dati di ogni argomento in una procedura Property Let e il suo corrispondente nella routine Property Get dovrebbe essere la stessa, fatta eccezione per l'ultimo argomento nella procedura Property Let che è supplementare. Nel caso di una procedura Property Let con un singolo argomento (è richiesto almeno un argomento da definire), questo argomento contiene il valore da assegnare alla proprietà ed è il valore fissato dalla procedura chiamante, in questo caso la procedura Property Get non avrà alcun argomento.

Una procedura Property Set può accettare più argomenti, e in questo caso l'ultimo argomento contiene il riferimento all'oggetto effettivo per la proprietà. Nel caso di una procedura Property Set con un singolo argomento (è richiesto almeno un argomento da definire), questo argomento contiene il riferimento all'oggetto per la proprietà. Il tipo di dati dell'ultimo argomento o il singolo argomento deve essere un tipo di oggetto o un valore Variant.

La procedura Property Set è simile e una variazione della procedura Property Let ed entrambi vengono utilizzati per impostare i valori. Una procedura Property Set viene utilizzata per creare le proprietà degli oggetti che sono in realtà puntatori ad altri oggetti, mentre una procedura Property Let assegna i valori alle proprietà scalari come stringhe, interi, date, etc.

Di seguito è riportata la sintassi per le dichiarazioni delle tre Procedura di proprietà.

Property Get
Property Get PropertyName(argomento_1, argomento_2, …, argomento_n) As Type

Property Let
Property Let PropertyName(argomento_1, argomento_2, …, argomento_n+1)

Property Set
Property Set PropertyName(argomento_1, argomento_2, …, argomento_n+1)

Esempio: Creare una classe Properties utilizzando procedure di proprietà. *Inserire il codice in un modulo di classe e denominarlo class_Stud*

```
Private nomeStud As String, votoStud As Double
Public Property Let Name(strN As String)
'si dichiara public in modo che possa essere chiamata da un'istanza in un altro modulo
    nomeStud = strN
End Property

Public Property Get Name() As String
'restituisce la proprietà Name
    Name = nomeStud
End Property

Public Property Let voto(ivoto As Double)
'assegna la proprietà voti
    votoStud = (ivoto / 80) * 100
End Property

Public Property Get voto() As Double
'restituisce la proprietà voti
    voto = votoStud
End Property

Public Function valuT() As String
'si dichiara public in modo che possa essere chiamata da un'istanza in un altro modulo.
    Dim valuta1 As String

    If votoStud >= 80 Then
        valuta1 = "A"
    ElseIf votoStud >= 60 Then
        valuta1 = "B"
    ElseIf votoStud >= 40 Then
        valuta1 = "C"
    Else
        valuta1 = "Bocciato"
    End If

valuT = valuta1
End Function
```

Inserire il codice sotto riportato in un modulo standard

```
Sub studenti()
'si utilizza l'istruzione Dim per creare una variabile e definirla come un riferimento alla classe.
    Dim studente As class_Stud
'si crea un nuovo riferimento a un oggetto utilizzando la parola chiave New. Indicare il nome
della classe a cui si desidera creare un'istanza, dopo la parola chiave New
Set studente = New class_Stud
    studente.Name = "Marco"
'si richiama la proprietà Name nell'oggetto studente
    MsgBox studente.Name
'si imposta la proprietà voto nell'oggetto studente al valore 45, e passa questi dati alla variabile
` i voti nella proprietà voto
studente.voto = 45
MsgBox studente.voto
MsgBox studente.valuT
MsgBox studente.Name & " ha ottenuto il " & studente.voto & " % di voti con valutazione " &
studente.valuT
End Sub
```

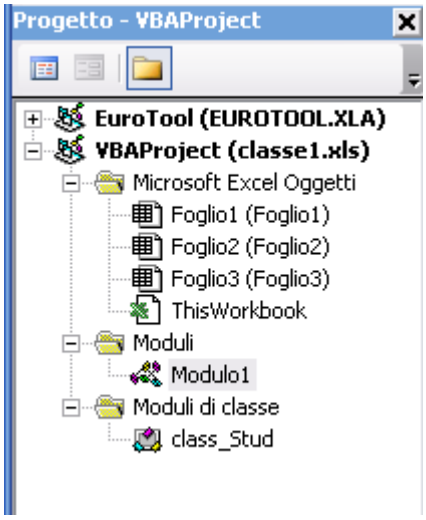

Fig. 1

Esempio: Creare una classe utilizzando la procedura Property Let passando più argomenti. *Inserire il codice in un modulo di classe e denominarlo class_rettangolo*

Nota: *Una procedura Property Let può accettare più argomenti, e in questo caso l'ultimo argomento contiene il valore da assegnare alla proprietà. Questo ultimo argomento nella lista degli argomenti è il valore della proprietà impostata dalla procedura chiamante. Il nome e il tipo di dati di ogni argomento in una procedura Property Let e il suo corrispondente la routine Property Get dovrebbe essere la stessa, fatta eccezione per l'ultimo argomento nella procedura Property Let che è opzionale.*

```
Private Are_a As Double
Public Property Let Area(Ba1 As Double, Alt1 As Double, ar As Double)
Are_a = ar
MsgBox "Argomenti ricevuti - Ba1: " & Ba1 & ", Alt1: " & Alt1 & ", ar: " & ar
End Property
Public Property Get Area(Ba1 As Double, Alt1 As Double) As Double
Area = Are_a
End Property
```

Inserire il codice sotto riportato in un modulo standard

```
Sub rettangolo()
Dim Ba As Double, Alt As Double
Dim rett As New class_rettangolo
Ba = InputBox("Inserire Base del Rettangolo")
Alt = InputBox("Inserire Altezza del Rettangolo")
rett.Area(Ba, Alt) = Ba * Alt
a = rett.Area(Ba, Alt)
MsgBox "L'area del Rettangolo con Base " & Ba & ", e Altezza " & Alt & ", è " & a
End Sub
```

Esempio: Creazione di sola lettura Classe proprietà con solo il blocco PropertyGet_EndProperty. *Inserire il codice in un modulo di classe e denominarlo classRettangolo*

```
Private rett_B As Double, rett_H As Double
Public Property Let Base(k As Double)
    rett_B = k
End Property

Public Property Get Base() As Double
    Base = rett_B
End Property

Public Property Let Altez(k1 As Double)
    rett_H = k1
End Property

Public Property Get Altez() As Double
    Altez = rett_H
End Property

Public Property Get area_r() As Double
    area_r = Base * Altez
End Property
```

Inserire il codice sotto riportato in un modulo standard

```
Sub rettangolo1()
Dim a As Double, b As Double
Dim areaRett As New classRettangolo
a = InputBox("Inserire Base rettangolo")
b = InputBox("Inserire Altezza rettangolo")

areaRett.Base = a
areaRett.Altez = b
MsgBox areaRett.area_r
End Sub
```

Esempio: Utilizzare Property Set per impostare un riferimento a un oggetto. *Inserire il codice in un modulo di classe e denominarlo classeAuto*

```
Private tipoVet As classeTipoAut
Public Property Set tipo(objtipo As classeTipoAut)
    Set tipoVet = objtipo
End Property

Public Property Get tipo() As classeTipoAut
    Set tipo = tipoVet
End Property
```

Inserire il codice sotto riportato in un modulo di classe e denominarlo classeTipoAut

```
Private tipoCol As String, tipoNom As String, KL As Double
Property Let colore(col As String)
    tipoCol = col
End Property
Property Get colore() As String
    colore = tipoCol
End Property
Property Let nome(nom As String)
    tipoNom = nom
End Property
Property Get nome() As String
    nome = tipoNom
End Property
Property Let percorso(kmLit As Double)
    KL = kmLit
End Property
Property Get percorso() As Double
    percorso = KL
End Property
Function costoBenz(cost_benz As Double, distanza As Double) As Double
    costoBenz = (distanza / percorso) * cost_benz
End Function
```

Inserire il codice sotto riportato in un modulo standard

```
Sub propSetCars()
Dim dist As Double, cost As Double
Dim auto As classeAuto
Set auto = New classeAuto
Set auto.tipo = New classeTipoAut
auto.tipo.colore = "Nero"
auto.tipo.nome = "Toyota Yaris"
auto.tipo.percorso = 50
dist = InputBox("Inserisci i chilometri percorsi in un mese dalla vettura")
cost = InputBox("Inserisci il costo del carburante al litro")
MsgBox "Il colore della vettura è : " & auto.tipo.colore
MsgBox "Il modello della vettura è : " & auto.tipo.nome
MsgBox "La tua auto percorre " & auto.tipo.percorso & " Km. con 1 litro di carburante"
MsgBox auto.tipo.costoBenz(dist, cost) & " Eur." & " è il costo mensile del carburante"
End Sub
```

Esempio: Utilizzo dell'istruzione Property Set per impostare un riferimento a un oggetto Range. Inserire il codice nel modulo di classe denominato classeRange:

```
Private coloreSF As Integer, SNome As String, vRng As Range
Public Property Set Range_At(oRng As Range)
   Set vRng = oRng
End Property

Public Property Get Range_At() As Range
   Set Range_At = vRng
End Property

Property Let Nome(nom As String)
   SNome = nom
End Property

Property Get Nome() As String
   Nome = SNome
End Property

Property Let colore(col As Integer)
   coloreSF = col
End Property

Property Get colore() As Integer
   colore = coloreSF
End Property

Sub Mcolore()
   Range_At.Interior.ColorIndex = colore
End Sub
```

Inserire il codice sotto riportato in un modulo standard

```
Sub classe_Range()
Dim RangeATT As classeRange
Set RangeATT = New classeRange
Set RangeATT.Range_At = ActiveCell
RangeATT.colore = 5

If RangeATT.colore < 1 Or RangeATT.colore > 56 Then
MsgBox "Errore! Inserisci un valore per il colore compreso tra 1 e 56"
Exit Sub
End If

RangeATT.Mcolore
MsgBox "colore di sfondo: ColorIndex " & RangeATT.colore & "  Inserito nella cella " &
RangeATT.Range_At.Address

End Sub
```

Eventi di Classe Personalizzati

Un codice VBA viene attivato quando si verifica un evento come, cliccare su un pulsante, aprire la cartella di lavoro, selezionare una cella o cambiare una selezione di celle in un foglio di lavoro, e così via. Excel ha anche le sue procedure di eventi che vengono attivati da un evento predefinito e vengono installati all'interno di Excel con un nome standard e predeterminato, come la procedura di modifica del foglio di lavoro che viene installato con il foglio di lavoro come "Private Sub Worksheet_Change (ByVal Target As Range)". Quando il contenuto di una cella del foglio di lavoro cambia, VBA chiama la routine evento Worksheet_Change ed esegue il codice che contiene. Ora vediamo come è possibile creare i propri eventi personalizzati in una classe.

Definire un evento personalizzato

Il primo passo è quello di dichiarare l'evento nella sezione di dichiarazione della classe usando la parola chiave **Event** per definire un evento personalizzato in un modulo di classe. Questa dichiarazione può avere qualsiasi numero di argomenti, e deve essere dichiarata come *Public* per renderlo visibile all'esterno del modulo oggetto. Si noti che è possibile dichiarare e generare eventi solo nei moduli oggetto ThisWorkbook, fogli di lavoro e fogli grafici, moduli Form e moduli di classe, e non da un modulo di codice standard.

Generare un evento

Dopo aver dichiarato un evento, si deve utilizzare una dichiarazione **RaiseEvent** per attivare l'evento dichiarato e la procedura di evento viene eseguita quando viene generato o attivato l'evento. Si ricorda che l'evento è dichiarato in un procedimento pubblico all'interno del modulo di classe utilizzando la parola chiave Evente con la dichiarazione *RaiseEvent* vengono passati i valori agli argomenti alla routine evento che viene eseguita

Codice esterno per generare l'evento

Abbiamo bisogno di un codice esterno per chiamare la procedura pubblica nel modulo di classe, che genera l'evento e tramite questo codice si determina quando l'evento verrà generato mediante il quale la procedura di evento viene eseguita.

Creare una routine evento

Si usa la parola chiave **WithEvents** per dichiarare una variabile oggetto della classe personalizzata in cui è definito l'evento personalizzato, dichiarando questa variabile oggetto, l'istanza della classe personalizzata punta alle variabili che risponderanno all'evento aggiungendo l'oggetto alla lista degli eventi nella finestra del codice. Solo le variabili dichiarate a livello di modulo possono essere utilizzato con la parola chiave WithEvents. Inoltre, le variabili possono essere dichiarate utilizzando le parole chiave WithEvents solo in moduli di oggetti e non in un modulo di codice standard. Dopo la dichiarazione della variabile oggetto, la procedura di evento può essere creata in modo simile alle procedure standard di VBA - la variabile oggetto verrà visualizzato nell'elenco a discesa Oggetti e tutti i suoi eventi sono elencati nell'elenco a discesa della procedura.

Esempio: Creare un evento personalizzato utilizzando una procedura Worksheet_Change per attivare l'evento personalizzato. Inserire il codice sotto riportato nel modulo di classe denominato classe_Range:

```
Private RngV As Range, coloreS As Integer, nomeS As String
Public Event selezionaC(cell As Range)

Public Property Set RangeS(objRng As Range)
    Set RngV = objRng
RaiseEvent selezionaC(RngV)
End Property

Public Property Get RangeS() As Range
Set RangeS = RngV
End Property

Property Let nome(nom As String)
nomeS = nom
End Property

Property Get nome() As String
nome = nomeS
End Property

Property Let colore(col As Integer)
coloreS = col
End Property

Property Get colore() As Integer
colore = coloreS
End Property

Sub AssColore()
RangeS.Interior.ColorIndex = colore
End Sub
```

Inserire il codice sotto riportato nel modulo del Foglio 1

```
Private WithEvents rang As classe_Range
Private Sub rang_selezionaC(cell As Range)
    rang.colore = 4
If rang.colore < 1 Or rang.colore > 56 Then
MsgBox "Errore! Inserire un valore valido per ColorIndex compreso tra 1 e 56"
Exit Sub
End If
rang.nome = "Prima Cella"

rang.AssColore
Dim i As Integer
i = rang.colore
rang.RangeS.Select
Selection.Offset(0, 1).Value = "Nome : " & rang.nome
Selection.Offset(0, 2).Value = "Indirizzo : " & Selection.Address
Selection.Offset(0, 3).Value = "Colore di sfondo : " & i
Selection.Offset(0, 4).Value = "Contenuto : " & Selection.Value
End Sub

Private Sub Worksheet_Change(ByVal Target As Range)
    On Error GoTo ErrorHandler
Set rang = New classe_Range
If Target.Address = Range("A1").Address Then
Set rang.RangeS = Target
Else
Exit Sub
End If

ErrorHandler:
  Application.EnableEvents = True

End Sub
```

Esempio: Creare un evento personalizzato: All'inizializzazione della Form si attiva l'evento personalizzato.

```
Private tb As MSForms.TextBox, strSeq As String
Public Event eTxtBx(objTxtBx As MSForms.TextBox)
Public Property Set Stesto(objTxtBx As MSForms.TextBox)
    Set tb = objTxtBx
RaiseEvent eTxtBx(tb)
End Property
Public Property Get Stesto() As MSForms.TextBox
    Set Stesto = tb
End Property
Property Let sequenza(tbSeq As String)
    strSeq = tbSeq
End Property
Property Get sequenza() As String
    sequenza = strSeq
End Property
```

Inserire una Form con 2 textBox (TextBox1 e TextBox2) e un CommandButton (CommandButton1) all'interno del modulo, inserire il seguente codice

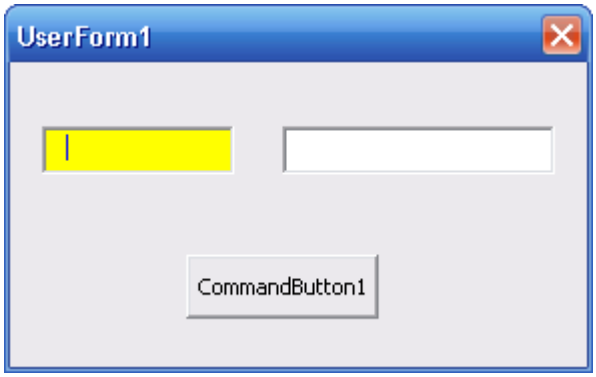

Fig. 2

```
Private WithEvents tx As classeTextBox, sq1 As String, sq2 As String
Private Sub CommandButton1_Click()
Dim objControl As Control
For Each objControl In Me.Controls
If TypeName(objControl) = "TextBox" Then
    If Not objControl.Name = "TextBox1" Then
    objControl.Value = "Valore Copiato : " & tx.Stesto.Value
    objControl.BackColor = vbRed
    End If
End If
Next
MsgBox "Testo Copiato dal " & sq1 & " Al " & sq2
End Sub
Private Sub TextBox1_Change()
    If tx.Stesto.Value = "" Then
        tx.Stesto.BackColor = vbYellow
    Else
        tx.Stesto.BackColor = vbGreen
    End If
End Sub
Private Sub tx_eTxtBx(objTxtBx As MSForms.TextBox)
tx.Stesto.BackColor = vbYellow
With Me.TextBox1
tx.sequenza = "Primo TextBox"
sq1 = tx.sequenza
End With
With Me.TextBox2
tx.sequenza = "Secondo TextBox"
sq2 = tx.sequenza
End With
End Sub
Private Sub UserForm_Initialize()
Set tx = New classeTextBox
Set tx.Stesto = Me.TextBox1
End Sub
```

Esempio: Creare un evento personalizzato: - utilizzare la parola chiave WithEvents, all'interno del modulo di classe, per dichiarare una variabile oggetto. Inserire il codice nel modulo di classe denominato classeCombo

```
Public WithEvents ComboB As MSForms.ComboBox
Public Sub SCombo(objCbx As MSForms.ComboBox)
    Set ComboB = objCbx
End Sub
Public Sub ComboB_AddItem(strItem As String, Cancel As Boolean)
    If Cancel = False Then
        ComboB.AddItem strItem
    End If
If strItem <> "" Then
    ComboB.BackColor = vbGreen
End If
End Sub
Private Sub ComboB_Change()
    If ComboB.Value = "" Then
        ComboB.BackColor = vbWhite
    End If
End Sub
```

Inserire una Form, con un ComboBox (ComboBox1) e un CommandButton (CommandButton1) all'interno del modulo. Inserire codice sotto riportato nel modulo della Form

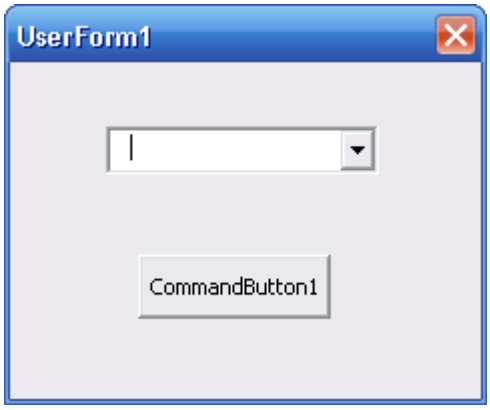

Fig. 3

```
Private cB As New classeCombo
Private Sub CommandButton1_Click()
Dim Stesto As String
    Stesto = cB.ComboB.Text
    Dim Cancel As Boolean
mess = MsgBox("Confermi di voler aggiungere la voce digitata al ComboBox?", vbYesNo)
If mess = vbNo Then
    Cancel = True
End If
Call cB.ComboB_AddItem(Stesto, Cancel)
End Sub
Private Sub UserForm_Initialize()
    cB.SCombo Me.ComboBox1
End Sub
```