I controlli ListBox e ComboBox

I controlli ListBox e ComboBox permettono di visualizzare una lista di opzioni selezionabili dall'utente. La ragione principale per usare una casella combinata (**ComboBox**) o una casella di riepilogo (**ListBox**) è quella di poter visualizzare un elenco di elementi, dove nella casella di riepilogo (ListBox) è un elenco a discesa e gli elementi della lista sono sempre visibili mentre nel ComboBox la lista è "a scomparsa", ovvero è visibile soltanto se l'utente fa clic sulla freccia verso il basso a destra del controllo. Le caselle combinate sono così chiamate perché sono composte da due parti, una porzione di testo che ricorda il controllo TextBox e un elenco che riporta ad una Listbox e "combinano" le caratteristiche che si trovano in entrambe le caselle di testo (TextBox) e le caselle di riepilogo (ListBox) e sono anche comunemente chiamate "elenchi a discesa".

In sostanza la ComboBox è un elenco a discesa in cui la voce selezionata dall'elenco è visibile nell'area di testo, mentre i valori della lista sono visibili solo cliccando sul menu a tendina, mentre un ListBox mostra un certo numero di valori con o senza una barra di scorrimento, inoltre in una ComboBox, è visibile una sola voce senza utilizzare la tendina a discesa, mentre in un ListBox sono visibili molti più elementi. In una ComboBox è possibile selezionare solo una voce dall'elenco, mentre invece in un ListBox è possibile selezionare più opzioni dall'elenco. La rappresentazione grafica delle due caselle è la seguente:

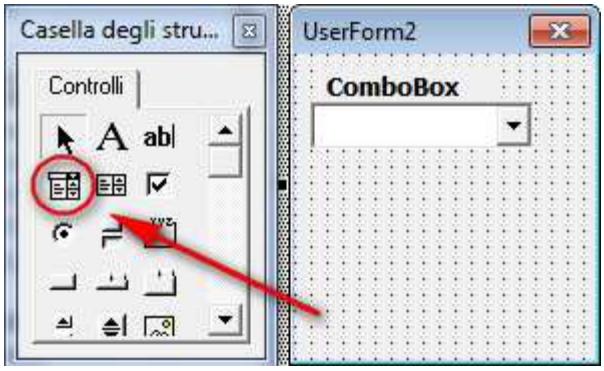

Fig. 1

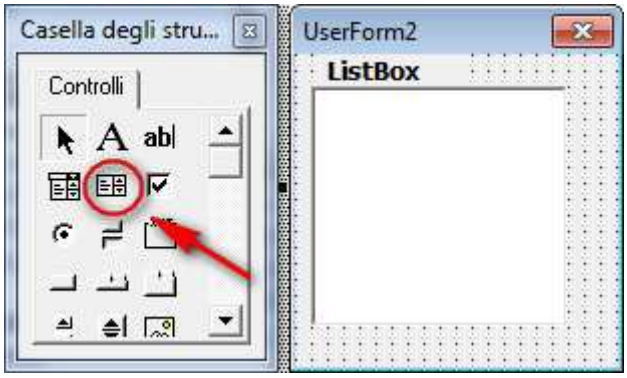

Fig. 2

Differenza tra ListBox e ComboBox

- In un ComboBox è visibile solo un elemento dell'elenco a discesa, e i valori della lista sono visibili utilizzando il menu a tendina, mentre un ListBox mostra un certo numero di valori con o senza una barra di scorrimento.
- In un ComboBox è possibile selezionare solo un'opzione dalla lista, mentre in un ListBox è possibile selezionare più opzioni dall'elenco.
- In un ComboBox è possibile inserire un valore digitandolo nell'area di testo se non è incluso nella lista, cosa che non è possibile fare in un ListBox.
- Il controllo CheckBox può essere utilizzato all'interno di un ListBox, ma non all'interno del ComboBox, inoltre il ListBox consente di visualizzare una casella di controllo accanto a ogni voce in elenco, per consentire all'utente di selezionare gli elementi. Per utilizzare il CheckBoxe in un controllo ListBox, si deve impostare la proprietà *ListStyle* nella finestra Proprietà su *fmListStyleOption* oppure utilizzando il codice VBA in questo modo:*ListBox1.ListStyle = fmListStyleOption*

Tutte le proprietà e i metodi indicati di seguito sono comuni a ListBox e ComboBox, salvo diversa indicazione, inoltre negli esempi di seguito riportati, i codici VBA devono essere inseriti nel modulo di codice del form, se non diversamente specificato.

Metodo AddItem:
Questo metodo permette di aggiunge una voce alla lista in un ListBox a colonna singola o in un ComboBox. La sintassi è la seguente: *Control.AddItem(Item, Index)*, che nel nostro caso diventa: *ListBox1.AddItem(Item, Index)*, dove *Item* specifica l'elemento o la riga da aggiungere e *Index* è un numero intero che specifica la posizione in cui il nuovo elemento è collocato all'interno della lista, se omesso, viene aggiunto l'elemento alla fine. I numeri di posizione o di riga iniziano con zero, e il primo elemento ha il numero 0, e così via. Il valore di indice non può essere maggiore del numero totale di righe. Il metodo*AddItem* può essere usato solo con un codice VBA. Per popolare un ComboBox è possibile utilizzare vari metodi, per esempio, se usiamo una casella combinata per permettere all'utente di vare una scelta "fissa" come può essere di scegliere una regione, un tipo di vettura etc. possiamo usare il metodo*AddItem* e impostare le varie voci direttamente dal codice, inoltre l'operazione di inserimento dati deve essere fatta nell'evento *Initialize* della Userfom in questo modo.

```
Private Sub UserForm_Initialize()
ComboBox1.Clear
ComboBox1.AddItem "Veneto"
ComboBox1.AddItem "Sardegna"
ComboBox1.AddItem "Lazio"
ComboBox1.ListIndex = 0
End Sub
```

Se mandiamo in esecuzione questo codice ci viene riportata questa finestra

Fig. 3

Da notare che la prima voce dell'enunciato è l'istruzione *ComboBox1.Clear* che serve per svuotare di eventuali dati rimasti in memoria del ComboBox in operazioni precedenti, per essere certi che la casella combinata è vuota, si può usare una condizione come questa : *If ComboBox1.ListCount >= 1 Then ComboBox1.Clear*

L'istruzione **ListCount** è una proprietà per determinare il numero di righe presenti in una casella combinata o di riepilogo, che inserendola in un enunciato IF verifica se nella casella sono presenti dei dati, infatti se *ListCount* è maggiore o uguale a 1 allora si esegue lo svuotamento del ComboBox tramite l'istruzione Clear, evitando così il rischio di trovarsi i dati ripetuti nella casella combinata. La penultima istruzione *ComboBox1.ListIndex = 0* Indica che si vuole far apparire la prima voce dell'elenco nel Combo come si vede in **figura 3**, se avessimo usato la dicitura *ComboBox.ListIndex = 1* avremmo come prima voce la seconda (Sardegna) presente nell'elenco. Possiamo utilizzare anche un ciclo per popolare il ComboBox in questo modo

```
Private Sub UserForm_Initialize()
 With ComboBox1
 .Clear
 .AddItem "Veneto"
 .AddItem "Sardegna"
 .AddItem "Lazio"
 .ListIndex =0
 End With
End Sub
```

Oppure se vogliamo popolare il ComboBox con i dati presenti in un Range del foglio di lavoro possiamo usare questo enunciato:

```
Private Sub UserForm_Initialize()
If ListBox1.ListCount >= 1 Then ListBox1.Clear
i = 1
Do Until Sheets("Foglio1").Cells(i, 1).Value = ""
With Sheets("Foglio1")
 elemento = .Cells(i, 1).Value
 End With
 ListBox1.AddItem elemento
 i = i + 1
 Loop
ListBox1.ListIndex = 0
End Sub
```

Per quanto riguarda il controllo ListBox non c'è nessuna differenza da quanto finora esposto, tutti i metodi e gli enunciati proposti possono essere usati anche per la casella di riepilogo ListBox avendo l'accortezza di sostituire il riferimento ComBox1 con ListBox1 (oppure il nome assegnato al controllo)

Metodo Clear

Questo metodo rimuove tutti gli elementi in un controllo ComboBox o ListBox e presenta la seguente Sintassi: *Control.Clear*, inoltre è da tenere presente che il metodo non funziona se ComboBox o ListBox viene associato a una fonte di dati utilizzando l'istruzione *RowSource*, in quanto i dati devono essere cancellati prima dell'uso del metodo Clear

Proprietà BoundColumn

Questa proprietà specifica la colonna da cui estrarre il valore da inserire in un controllo ComboBox o ListBox. La prima colonna ha un valore *BoundColumn* di 1, la seconda colonna ha un valore di 2, e così via, impostando il valore BoundColumn su 1 si assegnerà il valore della colonna 1 al ComboBox o ListBox permettendo così l'inserimento dei valori presenti nella colonna nel controllo. Impostando il valore di BoundColumn a 0 si assegna il valore della struttura*ListIndex*, che corrisponde al numero della riga selezionata, come valore del controllo ComboBox o ListBox. Questa impostazione è utile se si desidera determinare la riga della voce selezionata. La proprietà BoundColumn può essere impostata nella finestra Proprietà e può essere utilizzato anche con un codice VBA.

Esempio: Impostando il valore BoundColumn a 0 si assegna il valore della proprietà ListIndex come valore del controllo

```
Private Sub UserForm_Initialize()
With ListBox1
.ColumnHeads = True
.ColumnCount = 2
.ColumnWidths = "50;0"
.RowSource = "=Foglio2!A2:B6"
.MultiSelect = fmMultiSelectSingle
.TextColumn = 1
.BoundColumn = 0
End With
End Sub

Private Sub CommandButton1_Click()
If ListBox1.Value <> "" Then
TextBox1.Value = ListBox1.Value + 2
End If
End Sub
```

Proprietà Column

Si riferisce ad una colonna specifica, o combinazione di colonna e riga, in un ComboBox o ListBox a più colonne. Sintassi: Control.Column (iColumn, iRow). La proprietà *Column* può essere utilizzata solo con un codice VBA e non è disponibile in fase di progettazione. *iColumn* specifica il numero di colonna in cui la prima colonna della lista ha valore 0 e *iRow* specifica il numero di riga la prima riga della lista ha valore 0. Sia iColumn che iRow sono valori interi che vanno da 0 fino al numero di colonne e righe (rispettivamente) nell'elenco meno 1. Se si specifica entrambi i numeri di colonna e riga si farà riferimento ad un elemento specifico, mentre specificando solo il numero di colonna si farà riferimento a una colonna specifica. Per esempio [i]ListBox1.Column (1) è/i] si riferisce alla seconda colonna. È possibile copiare una matrice bidimensionale di valori in un controllo ListBox o ComboBox, utilizzando Column (o list) piuttosto che aggiungere ogni singolo elemento con il metodo AddItem.

Esempio: Popolare il ListBox utilizzando il metodo AddItem e List e utilizzare la proprietà Column per assegnare il contenuto di ListBox a TextBox

```
Private Sub UserForm_Initialize()
With ListBox1
.ColumnCount = 3
.ColumnWidths = "50;50;50"
.ColumnHeads = False
.RowSource = "=Foglio2!A2:B6"
.MultiSelect = fmMultiSelectMulti
End With
TextBox1 = ""
End Sub

Private Sub CommandButton1_Click()
'Il metodo AddItem non funziona se ListBox è associato ai dati, quindi la riga di origine
viene cancellata
ListBox1.RowSource = ""
'Crea una nuova riga con AddItem
ListBox1.AddItem "Banana"
'aggiunge un elemento nella seconda colonna della prima riga
ListBox1.List(0, 1) = "Martedì"
'aggiunta di elementi in 3 colonne della prima riga
ListBox1.List(0, 2) = "Giorno 2"
ListBox1.AddItem "Arancione"
'Aggiunge un elemento nella seconda colonna della seconda riga
ListBox1.Column(1, 1) = "Mercoledì"
'aggiunta di elementi in 3 colonne della seconda fila
ListBox1.Column(2, 1) = "Giorno 3"
ListBox1.AddItem "apple", 0
'Crea una nuova riga con AddItem e si posiziona nella riga 1
ListBox1.List(0, 1) = "Lunedì"
'Aggiunta di elementi in 3 colonne e posiziona questa riga come la prima
ListBox1.List(0, 2) = "Giorno 1"
'elemento in colonna 3 e numero di riga 2 del ListBox
TextBox1.Value = ListBox1.Column(2, 1)
End Sub
```

Proprietà ColumnCount

Specifica il numero di colonne da visualizzare in un controllo ComboBox o ListBox, un valore 0 di ColumnCount non visualizza nessuna colonna e questa proprietà può essere impostata nella finestra Proprietà oppure utilizzando un codice VBA.

Proprietà ColumnHeads

ColumnHeads restituisce un valore booleano (True o False) che determina la visualizzazione delle intestazioni di colonna per un ComboBox o ListBox e può essere impostata nella finestra Proprietà oppure utilizzando un codice VBA. Le intestazioni di colonna possono essere visualizzati solo se ColumnHeads è impostato su True nella finestra Proprietà oppure utilizzando il codice: *ListBox1.ColumnHeads = True*

Proprietà List

La proprietà *List* viene utilizzata in combinazione con *ListCount* e *ListIndex* per restituire gli elementi in un controllo ListBox o ComboBox. Sintassi :*Control.List (iRow, iCol)* . Ogni elemento in una lista ha un numero di riga e un numero di colonna, in cui i numeri di riga e di colonna iniziano da zero, iRow specifica il numero di riga e nel caso iRow = 2 rappresenta la terza riga della lista, mentre invece iColumn specifica il numero di colonna e nel caso che iColumn = 0 indica la prima colonna della lista, omettendo questo argomento iColumn recupererà la prima colonna. Si deve specificare iColumn solo per un ListBox o un ComboBox a più colonne. La proprietà List può essere utilizzata solo con un codice VBA e non è disponibile in fase di progettazione

Esempio: Utilizzare Selected, List per visualizzare e selezionare più elementi in un ListBox

```
Private Sub UserForm_Initialize()
With ListBox1
.ColumnHeads = True
.ColumnCount = 2
.ColumnWidths = "50;0"
.RowSource = "=Foglio3!A2:B6"
.MultiSelect = fmMultiSelectMulti
.TextColumn = 1
End With
With TextBox1
.MultiLine = True
.ControlSource = "=Foglio3!F2"
.Value = ""
End With
End Sub

Private Sub CommandButton1_Click()
TextBox1.Value = ""
For n = 0 To ListBox1.ListCount - 1
If ListBox1.Selected(n) = True Then
If TextBox1.Value = "" Then
TextBox1.Value = ListBox1.List(n, 1)
Else
TextBox1.Value = TextBox1.Value & vbCrLf & ListBox1.List(n, 1)
End If
End If
Next n
End Sub
```

Proprietà ListCount

Determina il numero totale di righe in un controllo ListBox o ComboBox, e questa proprietà può essere utilizzata solo con un codice VBA e non è disponibile in fase di progettazione.

Proprietà ListIndex

Determina quale elemento viene selezionato in un controllo ComboBox o ListBox. Il primo elemento di un elenco ha un valore di ListIndex uguale a 0, il secondo elemento ha un valorei 1, e così via, quindi, è un valore intero compreso tra 0 e il numero totale di elementi in un controllo ComboBox o ListBox meno 1. Questa proprietà può essere utilizzata solo con un codice VBA e non è disponibile in fase di progettazione. Nota: In una selezione multipla attivata in un ListBox, ListIndex restituisce l'indice della riga che ha il focus, a prescindere dal fatto che sia selezionata la riga o meno.

Proprietà ListRows

Specifica il numero massimo di righe che verranno visualizzate nella parte casella di riepilogo di un ComboBox. Il valore predefinito è 8. Nota: Se il numero effettivo di elementi della lista superi tale valore massimo della proprietà ListRows, apparirà una barra di scorrimento verticale nella casella di riepilogo del ComboBox e le voci in eccesso possono essere visualizzate scorrendo verso il basso). ListCount può essere utilizzata con la proprietà ListRows per specificare il numero di righe da visualizzare in un controllo ComboBox e può essere impostata nella finestra Proprietà oppure con un codice VBA. La proprietà ListRows è valida per ComboBox e non per ListBox.

Esempio: Utilizzare la proprietà ListCount e ListRows, per impostare il numero di righe da visualizzare in ComboBox

```
Private Sub UserForm_Initialize()
With ComboBox1
If .ListCount > 5 Then
.ListRows = 5
Else
.ListRows = .ListCount
End If
End With
End Sub
```

Proprietà MultiSelect

Specifica se sono consentite selezioni multiple e sono disponibili 3 impostazioni:
fmMultiSelectSingle (valore 0), è l'impostazione predefinita, in cui solo un singolo elemento può essere selezionato
fmMultiSelectMulti (valore 1), permette selezioni multiple in cui un elemento può essere selezionato o deselezionato facendo clic del mouse o premendo la barra spaziatrice
fmMultiSelectExtended (valore 2) permette selezioni multiple, in cui premendo il tasto SHIFT e contemporaneamente spostando la freccia su o giù (o premere MAIUSC e facendo clic del mouse) continua la selezione dalla voce precedentemente selezionata con la selezione corrente (cioè un continuo della selezione); questa opzione permette anche di selezionare o deselezionare una voce premendo CTRL e cliccando il mouse.

La proprietà *MultiSelect* può essere impostata nella finestra Proprietà oppure con un codice VBA. Nota: La proprietà MultiSelect è valida per ListBox e non per ComboBox . Quando vengono effettuate selezioni multiple, gli elementi selezionati possono essere determinati solo mediante la proprietà Selected immobili della ListBox. La struttura Selected avrà valori che vanno da 0 a ListCount meno 1 e sarà True se l'elemento è selezionato e False se non selezionata.

Esempio: Determinare l'elemento selezionato in un controllo ListBox a selezione singola

```
Private Sub CommandButton1_Click()
If ListBox1.Value <> "" Then
MsgBox ListBox1.Value
End If
End Sub
```

Metodo RemoveItem

Viene utilizzato per rimuovere una riga dalla lista in un ComboBox o ListBox.
Sintassi: *Control.RemoveItem (row_index)*, dove *Row_index* è il numero di riga che deve essere rimosso, considerando che la prima riga ha valore 0. Il metodo RemoveItem non funziona se ComboBox o ListBox viene associato ai dati, quindi utilizzando RowSource i dati devono essere cancellati prima dell'uso, inoltre questo metodo può essere utilizzato solo con un codice VBA.

Proprietà RowSource

Specifica la riga di origine di una lista (che potrebbe essere un intervallo del foglio di lavoro), per un ComboBox o ListBox. La proprietà RowSource può essere impostata nella finestra Proprietà oppure con un codice VBA. Per impostare RowSource nella finestra delle Proprietà, digitare senza virgolette il Range di origine in questo modo: " = *Foglio2 A2: A6*" che popola il ComboBox o il ListBox con i valori presenti nelle celle A2: A6 del Foglio2. Il codice VBA per questo passaggio è: *ListBox1.RowSource = "= Foglio2 A2: A6"*. Non è necessario utilizzare il segno di uguale ("= Foglio2 A2: A6"), impostando la proprietà con *ListBox1.RowSource = "Foglio2 A2: A6"* avrà lo stesso effetto.

Proprietà Selected

Specifica se un elemento viene selezionato in un controllo ListBox. Sintassi: *Control.Selected (Item_Index)* e restituisce Vero o Falso, se l'elemento è selezionato oppure no, impostando a True per selezionare la voce o rimuovere la selezione [vale a dire. Control.Selected (Item_Index) = True]. Item_Index è un valore intero compreso tra 0 e il numero di elementi nella lista meno 1, indicando la sua posizione relativa nella lista, vale a dire *ListBox.Selected (2) = True* seleziona il terzo elemento della lista. Questa proprietà è particolarmente utile quando si lavora con selezioni multiple e può essere utilizzata solo con un codice VBA e non è disponibile in fase di progettazione. Nota: In una selezione multipla in un ListBox, ListIndex restituisce l'indice della riga che ha il focus, a prescindere dal fatto che sia selezionata la riga o meno, da cui la proprietà Selected della ListBox può essere utilizzata per impostare una nuova selezione. In una selezione standard del ListBox, ListIndex restituisce l'indice dell'elemento selezionato e quindi dovrebbe essere usata per impostare una nuova selezione.

Esempio: Determinare gli elementi selezionati in una selezione multipla ListBox utilizzando selected e List

```
Private Sub CommandButton1_Click()
For n = 0 To ListBox1.ListCount - 1
If ListBox1.Selected(n) = True Then
MsgBox ListBox1.List(n)
End If
Next n
End Sub
```

Proprietà Style

Questa proprietà è valida per ComboBox e non per ListBox e determina se la scelta della voce può essere fatta dalla lista a discesa oppure digitando nel campo di testo il valore della ComboBox. Dispone di due impostazioni:

- *fmStyleDropDownCombo* (valore 0). L'utente ha entrambe le opzioni di digitare un valore personalizzato nell'area di testo o selezionare dall'elenco a discesa. Questo è il valore predefinito.
- *fmStyleDropDownList* (valore 2). L'utente può selezionare solo dall'elenco a discesa, come in ListBox.

La proprietà *Style* può essere impostata nella finestra Proprietà oppure con codice VBA.

Proprietà TextColumn

Specifica la colonna da visualizzare in un controllo ComboBox o ListBox a più colonne, quando una riga è selezionata dall'utente. La prima colonna ha un valore TextColumn di 1, la seconda ha un valore di 2, e così via. Impostare il valore di TextColumn a 0 visualizza il valore ListIndex (che è il numero della riga selezionata) nella proprietà TextColumn, questa impostazione è utile se si desidera determinare la riga della voce selezionata.
La proprietà *TextColumn* può essere impostata nella finestra Proprietà o con codice VBA. Nota: In un ComboBox, quando un utente seleziona un elemento, la colonna specificata nella proprietà TextColumn verrà visualizzato nella parte casella di testo del ComboBox.

Esempio: Utilizzo di TextColumn per visualizzare prima colonna e BoundColumn per la seconda colonna in una ListBox a selezione singola

```
Private Sub UserForm_Initialize()
With ListBox1
.ColumnHeads = True
.ColumnCount = 2
.ColumnWidths = "50;0"
.RowSource = "=Foglio2!A2:B6"
.MultiSelect = fmMultiSelectSingle
.TextColumn = 1
.BoundColumn = 2
End With
End Sub

Private Sub CommandButton1_Click()
If ListBox1.Value <> "" Then
TextBox1.Value = ListBox1.Value & " cms"
End If
End Sub
```

Aggiungere elementi per popolare un controllo ListBox o ComboBox

Impostare la proprietà RowSource di un controllo ListBox o ComboBox in un form utente per un elenco statico:
Me.ListBox1.RowSource = "Foglio1 A1: B6" oppure *Me.ListBox1.RowSource = "= Foglio1 A1: B6"*
Mentre per un elenco dinamico: *Me.ListBox1.RowSource = "Foglio1 A1: B" . & Foglio1.Cells (Rows.Count, "B") End (xlUp) Row*

Esempio: Popolare un ComboBox impostando la proprietà RowSource di una lista denominata

```
Private Sub UserForm_Initialize()
With ComboBox1
.ColumnCount = 2
.ColumnWidths = "50;50"
.ColumnHeads = True
.RowSource = "Foglio1!nome_intervallo"  <<<< da cambiare con Vs. nome intervallo
End With
End Sub
```

Popolare un ComboBox o ListBox da un array a colonna singola in un ListBox:
ListBox1.List = Array ("Riga1", "Riga2", "Riga3", "Riga4")

in un ComboBox:
ComboBox1.List = array ("Riga1", " Riga2", " Riga3", " Riga4")

Compilare un ListBox da una matrice denominata myArray:
Dim myArray As Variant
myArray = Array ("Adidas", "Nike", "Reebok")
Me.ListBox1.List = myArray

Popolare una singola colonna in un ComboBox:
Dim i As Integer
Dim myArray As Variant
myArray = Array ("Adidas", "Nike", "Reebok", "Puma", "Polo")
For i = LBound(myArray) To UBound(myArray)
Me.ComboBox1.AddItem myArray(i)
Next

Esempio: Popolare un ListBox a più colonne da un Range del foglio di lavoro

```
Private Sub UserForm_Initialize()
With ListBox1
.ColumnCount = 3
.ColumnWidths = "50;50;50"
.ColumnHeads = False
End With
Dim rng As Range
Set rng = Foglio1.Range("A1:C6")
Me.ListBox1.List = rng.Cells.Value
End Sub
```

Esempio: Popolare un ListBox a più colonne dal Range del Foglio di lavoro

```
Private Sub UserForm_Initialize()
With ListBox1
.ColumnCount = 3
.ColumnWidths = "50;50;50"
.ColumnHeads = False
End With
Dim var As Variant
var = Foglio1.Range("A1:C6")
Me.ListBox1.List = var
End Sub
```

Esempio: Popolare un ListBox a più colonne dal Range di un foglio di lavoro dopo aver posizionato i dati di in una matrice a 2 dimensioni

Codice:

```
Option Base 1
Private Sub UserForm_Initialize()
Dim rng As Range
Dim cell As Range
Dim totalRiga As Integer, totalColon As Integer
Dim iRow As Integer, iCol As Integer
Dim myArray() As Variant
Set rng = Foglio1.Range("A1:C6")
totalRiga = Foglio1.Range("A1:C6").Rows.Count
totalColon = Foglio1.Range("A1:C6").Columns.Count
ReDim myArray(totalRiga, totalColon)
For Each cell In rng
For iRow = 1 To totalRiga
For iCol = 1 To totalColon
myArray(iRow, iCol) = rng.Cells(iRow, iCol)
Next iCol
Next iRow
Next
With ListBox1
.ColumnCount = 3
.ColumnWidths = "50;50;50"
.ColumnHeads = False
.List = myArray
End With
End Sub
```

Esempio: Caricare una matrice a 2 dimensioni in un ListBox utilizzando la proprietà List

Codice:

```
Private Sub UserForm_Initialize()
With ListBox1
.ColumnCount = 3
.ColumnWidths = "50;50;50"
.ColumnHeads = False
End With
With ListBox2
.ColumnCount = 3
.ColumnWidths = "50;50;50"
.ColumnHeads = False
End With
End Sub
```

```
Private Sub CommandButton1_Click()
Dim myArray(3, 3)
For n = 0 To 2
myArray(n, 0) = n + 1
Next n
myyArray(0, 1) = "R1C2"
myArray(1, 1) = "R2C2"
myArray(2, 1) = "R3C2"
myArray(0, 2) = "R1C3"
myArray(1, 2) = "R2C3"
myArray(2, 2) = "R3C3"
ListBox1.List() = myArray
ListBox2.Column() = myArray
End Sub
```

Esempio: Popolare un ComboBox con i 12 mesi in un anno
Codice:

```
Private Sub UserForm_Initialize()
With ComboBox1
.ColumnCount = 1
.ColumnWidths = "50"
.ColumnHeads = False
.RowSource = ""
End With
For n = 1 To 12
ComboBox1.AddItem Format(DateSerial(2014, n, 1), "mmmm")
Next n
End Sub
```

Esempio: Popolare una ListBox a più colonne da un Range utilizzando il metodo AddItem e List
Codice:

```
Private Sub UserForm_Initialize()
With ListBox1
.ColumnCount = 3
.ColumnWidths = "50;50;50"
.RowSource = ""
End With
End Sub

Private Sub CommandButton1_Click()
Dim conta As Long
Dim totalRiga As Long
totalRiga = Foglio4.Cells(Rows.Count, "A").End(xlUp).Row
conta = 0
Do
With Me.ListBox1
conta = conta + 1
.AddItem Foglio4.Cells(conta, 1).Value
.List(.ListCount - 1, 1) = Foglio4.Cells(conta, 1).Offset(0, 1).Value
.List(.ListCount - 1, 2) = Foglio4.Cells(conta, 1).Offset(0, 2).Value
End With
Loop Until conta = totalRiga
End Sub
```

```
Dim rng1 As Range, rng2 As Range, rngUnion As Range
'impostare un blocco contiguo di celle come primo intervallo (o Range)
Set rng1 = Range ("A1: B2")
'impostare un altro blocco contiguo di celle come secondo intervallo
Set rng2 = Range ("D3: E4")
'assegnare una variabile (oggetto range) per rappresentare l'unione dei due intervalli
Set rngUnion = Union (rng1, rng2)
'colore interno impostato per l'intervallo che è l'unione di 2 oggetti Range
rngUnion.Interior.Color = vbYellow
```