Utilizzare il controllo ListView

Il controllo **ListView** viene utilizzato per visualizzare informazioni di tipo gerarchico e consente di mostrare elenchi o liste di dati, oltre ad essere di grande impatto visivo. Questo tipo di controllo è diventato popolare con Windows Explorer in quanto è lo stesso tipo di lista usata dall'interfaccia di Explorer (Gestione risorse o Esplora risorse) per visualizzare Files e cartelle, inoltre le sue proprietà permettono di personalizzarne la visualizzazione in quattro stili diversi che sono:

- *LargeIcon* (Icone grandi): Ogni elemento è visualizzato con un'icona e del testo in basso
- *SmallIcon* (Icone piccole): Ogni elemento è visualizzato con una piccola icona e del testo alla sua destra.
- *Details* (Dettagli): Gli elementi sono disposti uno per riga. Ogni riga è suddivisa in più colonne, la prima contiene l'elemento stesso, le altre visualizzano i suoi attributi, inoltre ogni colonna ha un'intestazione.
- *List* (Elenco): Ogni elemento è visualizzato con una piccola icona e del testo alla sua destra e gli elementi sono organizzati in colonne senza intestazione.

Per rendersi conto di come viene visualizzata ciascuna modalità basta aprire Windows Explorer e selezionare i comandi corrispondenti nel menu Visualizza e per darvi un'idea della flessibilità di questo controllo dovete sapere che il desktop di Windows non è nient'altro che un grande controllo ListView in modalità Icon, con sfondo trasparente.
Il controllo ListView è un componente aggiuntivo **.ocx** che può essere aggiunto a Visual Basic attraverso Windows Common Control 6.0 e per averlo disponibile in VB per Excel si deve seguire il percorso dal menu **Strumenti – Controlli aggiuntivi** e nella finestra che appare scorrere la lista e selezionare la voce **Microsoft Listview Control, versione 6.0**.

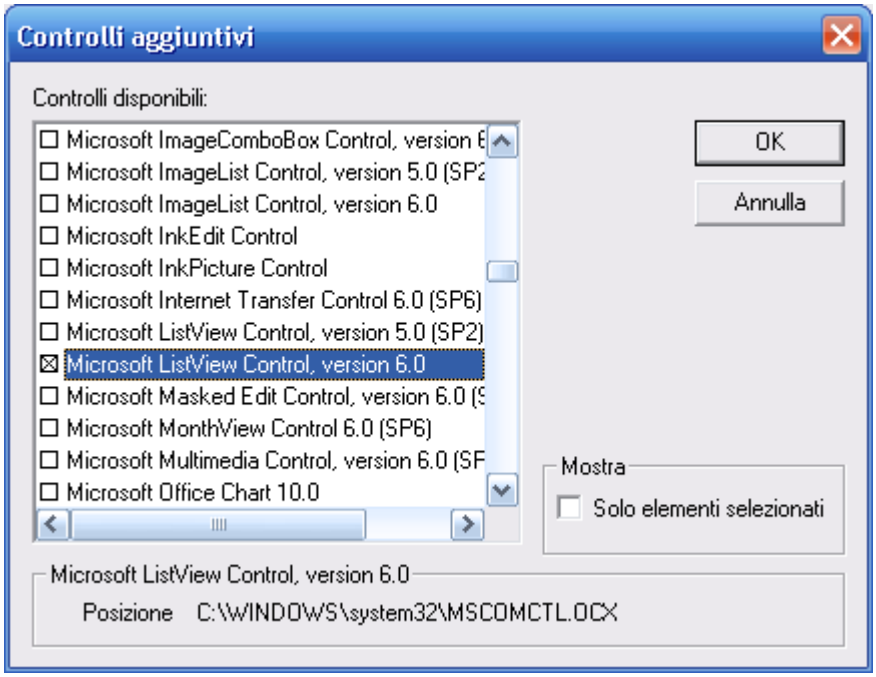

Fig. 1

Una volta confermata la scelta premendo sul pulsante Ok nella casella degli strumenti apparirà l'icona del controllo ListView

Fig. 2

Il controllo ListView dispone di due collection distinte:

- *ListItems* che comprende gli elementi testuali e grafici che rappresentano le voci da visualizzare e in modalità Report, è possibile specificare, per ogni voce, tutte le rispettive sottovoci mediante l'array ListSubItems.
- *ColumnHeaders* che include oggetti che influenzano l'aspetto delle singole intestazioni delle colonne visibili in modalità Report.

Esaminiamo ora la proprietà ListView che può essere: Icon View, Small Icon View, List View e Report View e per impostare una modalità di visualizzazione si deve assegnare alla proprietà View uno dei seguenti valori costanti:

0-lvwIcon = Icon View
1-lvwSmallIcon = Small Icon View
2-lvwList = List View
3-lvwReport = Report View

Per aggiungere un elemento al controllo ListView si deve usare il **metodo Add()** della collezione *ListItems*, tenendo presente che ogni riga di un oggetto ListView può essere definito in due parti

ListView1.ListItems(x): Specifica la riga per la prima colonna della riga
ListView1.ListItems(x).ListSubItems(y): Consente di specificare le colonne adiacenti, per esempio: *ListView1.ListItems(5).ListSubItems(1)* indica la seconda colonna della quinta riga della ListView

Per cui la sintassi per aggiungere **una riga** è la seguente: *ListView1.ListItems.Add [Index], [Key], [Text], [Icon], [SmallIcon]* , in cui le voci rappresentano:

[Index]: Valore Opzionale
È un numero che indica la posizione della voce all'interno della collezione. Se viene omesso, la voce viene inserita in coda alla collezione. Il suo valore varia in funzione di successivi inserimenti e cancellazioni
[Key]: Valore Opzionale
E' una stringa che rappresenta la chiave identificativa univoca di ogni voce nella collezione. Utile per la ricerca di una voce.
[Text]: Valore Opzionale
Rappresenta il testo che viene mostrato nel controllo, eventualmente associata ad una icona.
[Icon]: Valore Opzionale
Specifica l'immagine da visualizzare quando il ListView è la modalità lvwIcon
[SmallIcon]: Valore Opzionale
Specifica l'immagine da visualizzare quando ListView è lvwSmallIcon, lvwList o in modalità lvwReport

Mentre invece la sintassi per aggiungere un elemento alla colonna di destra della riga specificata è la seguente: *ListView1.ListItems(1).ListSubItems.Add [Index], [Key], [Text], [ReportIcon], [TooltipText]* , dove:

1:
Specifica il numero della riga nel controllo ListView.
[Index]: Valore Opzionale
Specifica il numero di colonna per l'aggiunta di un dato. Il valore 1 corrisponde alla seconda colonna di un oggetto ListView
[Key]: Valore Opzionale
E' una stringa che rappresenta la chiave identificativa univoca di ogni voce nella collezione
[Text]: Valore Opzionale
Specifica il testo che verrà visualizzato nel ListView
[ReportIcon]: Valore Opzionale
Visualizza un'icona o un'immagine in base all'elemento specificato
[TooltipText]: Valore Opzionale
Aggiunge un tooltip nell'elemento specificato

Mentre invece per aggiungere delle **colonne**, è necessario prima definire i testi, le dimensioni e le intestazioni usando la seguente sintassi:*ListView1.ColumnHeaders.Add [Index], [Key], [Text], [Width], [Alignment], [Icon]*, dove le voci rappresentano:

[Index]: Valore Opzionale
[Key] Valore Opzionale .
E' una stringa che rappresenta la chiave identificativa univoca di ogni voce nella collezione
[Text]: Valore Opzionale .
Specifica il testo che verrà visualizzato nel ListView.
[Width]: Valore Opzionale .
Specifica la larghezza della colonna. Il valore predefinito è 72 punti
[Alignment]: Valore Opzionale .
Specifica l'allineamento della colonna. Le costanti disponibili:. LvwColumnLeft (Default) lvwColumnCenter, lvwColumnRight
[Icon]: Valore Opzionale .
Specifica l'immagine da visualizzare nell'intestazione.

L'esempio che segue mostra il principio di riempimento di un oggetto ListView.

```
Private Sub UserForm_Initialize()
    With ListView1
        'Imposta il numero di colonne e intestazioni
        With .ColumnHeaders
            'Rimuove le vecchie intestazioni
            .Clear
     'aggiunge 3 colonne specificando il nome dell'intestazione e la larghezza della colonna
            .Add , , "Nome", 80
            .Add , , "Città", 50
            .Add , , "Età", 50
        End With
       'Riempie la prima colonna con 3 righe
        With .ListItems
           .Add , , "Gino"
           .Add , , "Mario"
           .Add , , "Elisa"
         End With
      'Riempie le colonne 2 e 3 della prima riga
        .ListItems(1).ListSubItems.Add , , "Città1"
        .ListItems(1).ListSubItems.Add , , 30
       'Riempie le colonne 2 e 3 della seconda riga
        .ListItems(2).ListSubItems.Add , , "Città2"
        .ListItems(2).ListSubItems.Add , , 27
        'Riempie le colonne 2 e 3 della terza riga
        .ListItems(3).ListSubItems.Add , , "Città3"
        .ListItems(3).ListSubItems.Add , , 41
    End With
  'Specifica le modalità di visualizzazione in "Dettagli"
    ListView1.View = lvwReport
End Sub
```

E si ottiene una Form come la seguente

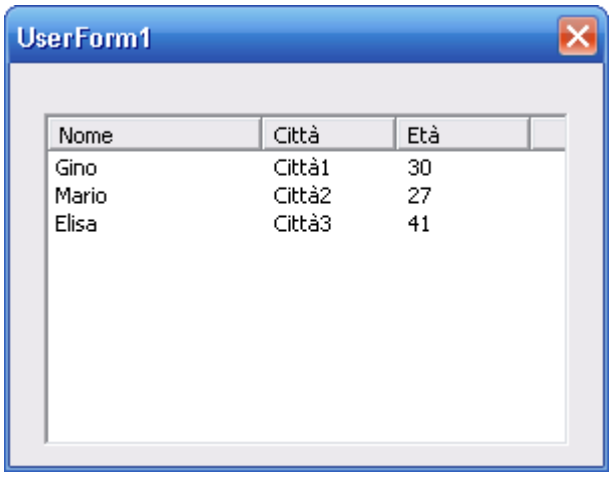

Fig. 3

Questa macro è un esempio semplificato ed è ovviamente possibile creare un loop per ottimizzare il riempimento. Una volta visualizzati nel controllo, i dati possono essere letti e modificati, la procedura sotto riportata scorre tutto il ListView e trasferisce le informazioni in un foglio di calcolo.

```
Private Sub CommandButton1_Click()
    Dim i As Integer, j As Integer
   'ciclo su tutte le righe
    For i = 1 To ListView1.ListItems.Count
        Cells(i, 1) = ListView1.ListItems(i).Text
       'Loop sulle colonne
        For j = 1 To ListView1.ColumnHeaders.Count - 1
            Cells(i, j + 1) = ListView1.ListItems(i).ListSubItems(j).Text
        Next j
    Next i
End Sub
```

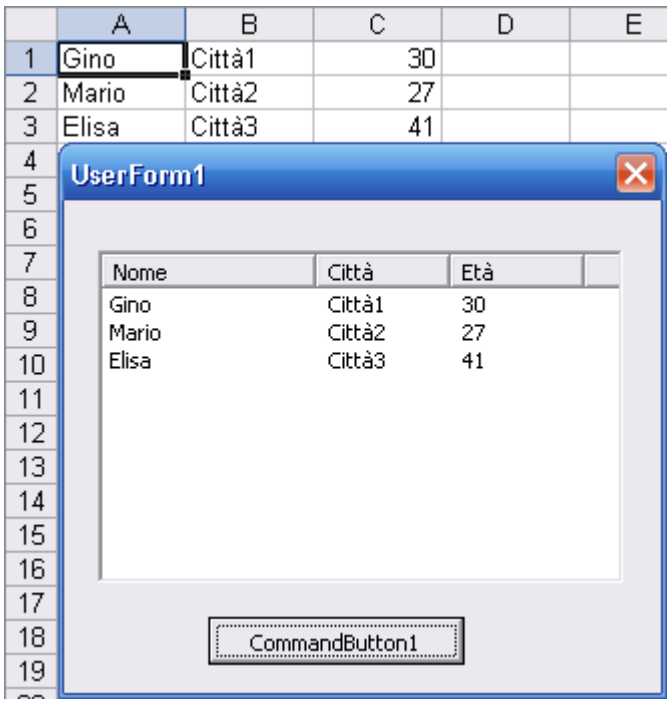

Fig. 4

Le informazioni contenute in una ListView possono essere facilmente modificate, ad esempio, se vogliamo modificare il testo nella terza colonna della prima riga possiamo usare un codice come il seguente

```
ListView1.listItems(1).listSubItems(2).Text = "Prova"
```

Un altro esempio per cambiare il testo nella terza riga della prima colonna

```
ListView1.ListItems(3).Text = "prova Modifica"
```

I dati nella prima colonna possono essere modificati anche manualmente nel controllo ListView, ma è possibile impedire questa azione specificando il valore 1 (lvwManual) nella **proprietà LabelEdit**. Oppure tramite questo codice

```
ListView1.labeledit = 1
```

E' inoltre possibile assegnare elementi chiave unici in una ListView in modo che i dati possono essere recuperati tramite questa chiave di identificazione. Per esempio può essere recuperato il contenuto della voce a cui viene assegnato il tasto "A1". Nota: la procedura restituisce un errore se la chiave non esiste nel ListView.

```
MsgBox ListView1.ListItems("A1").Text
```

E per recuperare una specifica voce sotto "A2" nell'elemento "A1"

```
MsgBox ListView1.ListItems("A1").ListSubItems("A2").Text
```

È inoltre possibile ottenere la chiave di una riga, questa procedura restituisce una stringa vuota se non è stato assegnato nessun tasto.

```
MsgBox ListView1.ListItems(2).Key
```

Questa macro consente di assegnare una chiave per la ListItem della seconda riga e se la chiave esiste già per questo elemento, verrà sovrascritto. Se si tenta di assegnare una chiave già assegnata ad un altro elemento, la procedura restituisce un messaggio di errore, e questo è logico in quanto la chiave deve essere univoca.

```
Private Sub CommandButton2_Click()
   'Legge la chiave originale
    MsgBox ListView1.ListItems(2).Key
   'Assegnare una nuova chiave al ListItem della seconda riga
    ListView1.ListItems(2).Key = "Nuova Key"
    'verifica  delle  nuove  chiavi
    MsgBox ListView1.ListItems(2).Key
End Sub
```

Inoltre è possibile eliminare delle righe specifiche nella ListView.

```
  'Rimuovere la terza riga
ListView1.ListItems.Remove 3
'altro esempio per eliminare una riga in base alla sua Key
ListView1.ListItems.Remove "A1"
'Eliminare la riga attiva
ListView1. ListItems. Remove (ListView1. SelectedItem. Index)
```

Mentre invece per eliminare tutti i dati in un controllo ListView si usa questo codice:

```
  ListView1.ListItems.Clear
```

E possibile modificare la formattazione del testo del controllo per personalizzare la visualizzazione delle informazioni, nel codice sotto riportato si modifica il colore del testo nel 2 ° elemento della prima riga.

```
ListView1.listitems(1).ListSubItems(2).ForeColor = RGB(100, 0, 100)
```

Oppure si può applicare un formato a una "cella" del listview

```
ListView1.ListItems(2).ListSubItems.Add , , Format(1234567.89, "##,##0.00")
```

Inoltre tramite la proprietà *FullRowSelect* si può evidenziare l'intera riga in una selezione.

```
ListView1.FullRowSelect = True
```

La proprietà *Griglia* permette di visualizzare una griglia nel ListView, questa proprietà è molto utile per migliorare la leggibilità dei dati

```
ListView1.Gridlines = True
```

Un'altra opzione del controllo consente di visualizzare le caselle di controllo nella colonna di sinistra.

```
Me.ListView1.CheckBoxes = True
```

È quindi possibile specificare lo stato di default del CheckBox, se non si specifica questo parametro, il testo non sarà visibile subito, ma è necessario fare clic sul bordo sinistro della riga per far apparire il CheckBox.

```
Dim i As Integer
For i = 1 To ListView1.ListItems.Count
   ListView1.ListItems(i).Checked = False
Next i
```

Possiamo usare l'evento *ItemCheck* per identificare quando una casella è selezionata oppure deselezionata e modificare il colore del testo in blu e grassetto.

```
Private Sub ListView1_ItemCheck(ByVal Item As MSComctlLib.ListItem)
   Dim j As Integer
   If Item.Checked = True Then
         'Cambia Colore
         Item.ForeColor = RGB(0, 0, 255)
         'Imposta il grassetto
         Item.Bold = True

         For j = 1 To Item.ListSubItems.Count
            Item.ListSubItems(j).ForeColor = RGB(0, 0, 255)
            Item.ListSubItems(j).Bold = True
         Next j
      Else
          'Cambia Colore
         Item.ForeColor = RGB(1, 0, 0)
         Item.Bold = False

         For j = 1 To Item.ListSubItems.Count
            Item.ListSubItems(j).ForeColor = RGB(1, 0, 0)
            Item.ListSubItems(j).Bold = False
         Next j
   End If
End Sub
```

Inoltre è possibile scegliere di nascondere le intestazioni delle colonne utilizzando la proprietà *HideColumnHeaders*.

```
ListView1.HideColumnHeaders = True
```

La seguente macro specifica che i dati devono essere centrati nella colonna

```
ListView1.ColumnHeaders.Add , , "Città", 50, lvwColumnCenter
```

La struttura permette tramite la proprietà *AllowColumnReorder* di spostare la posizione delle colonne tramite un drag & drop.

```
ListView1.AllowColumnReorder = True
```