[Access]

Michele de Nittis

Tutorial – ActiveX - La Funzionalità Drag – And – Drop Del Controllo Tree View

---

Versione: 1

Data Versione: lunedì 16 maggio 2005

Sommario

Premessa

Nel presente documento verrà descritta la funzionalità **Drag –and – Drop** relativamente ai controlli *Activex TreeView Control 6.0* e *ListView Control 6.0*. Si esaminerà prima il caso di trascinamento di un nodo di un controllo *TreeView* all'interno del controllo medesimo, poi il caso di trascinamento di un *Item* da un controllo *ListView* verso un controllo *TreeView*. Per semplicità espositiva si farà riferimento ad un esempio pratico di assegnazione di entità **Studenti** ad entità **Professore** con le seguenti semplici regole:

1) Ogni **Studente** può essere assegnato al più ad un solo **Professore**;
2) Un **Professore** non può essere assegnato ad un altro **Professore**;
3) Un **Professore** non può essere assegnato ad uno **Studente**.

Visualizzeremo le relazioni **Professore-Studente** in un albero (controllo *TreeView*) a due livelli: al primo livello saranno associati i **Professori**, al secondo gli **Studenti**. Gli studenti non ancora associati ad un **Professore** verranno visualizzati in un apposito elenco (controllo *ListView*).

Limiteremo, per semplicità, le operazioni di associazione alle seguenti:

Op.1) Nuova assegnazione di uno **Studente** da un **Professore** ad un altro **Professore**;

Op.2) Prima assegnazione di uno **Studente** ad un **Professore**.

Preparazione dei Controlli

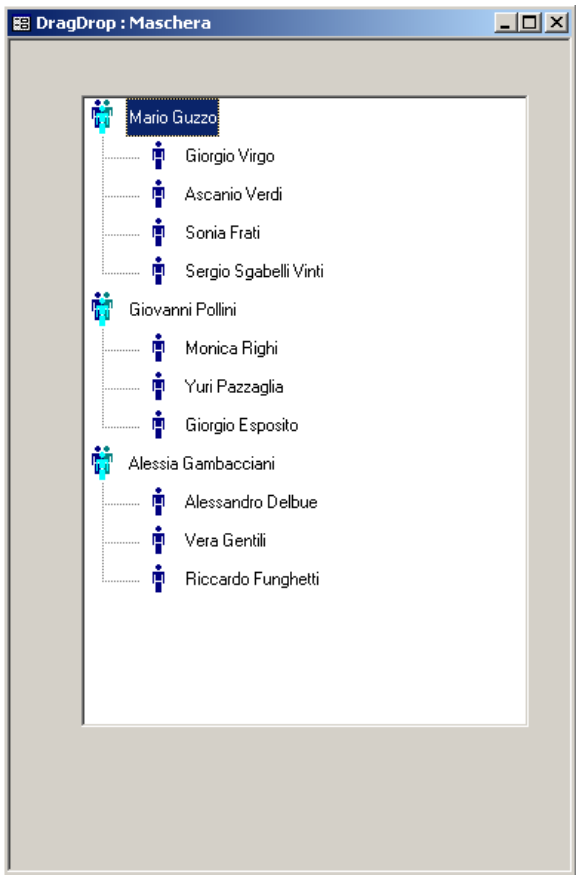
Inseriamo il controllo Tree View 6.0 all'interno di una maschera vuota e creiamo un albero di prova per testare la funzionalità Drag – Drop. Per ulteriori dettagli vedere "*Tutorial – ActiveX – Control Tree View* di Michele de Nittis [1].

Chiamiamo il controllo *Tree View* **R_AlberoTest** e la maschera che lo contiene **ProvaDragDrop**. Inseriamo, inoltre, nella maschera un controllo di tipo **ImageList** in modo da migliorare l'aspetto grafico dell'albero. Per ulteriori dettagli vedere "*Tutorial – Active X – Oggetto Imagelist ….*" di Michele de Nittis [2].

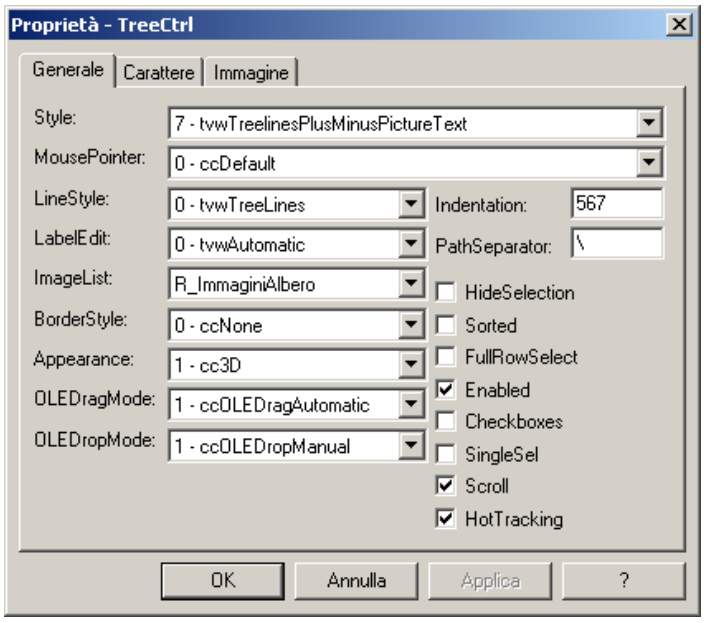
Impieghiamo l'evento ***Form_Load()*** della maschera **ProvaDragDrop** per popolare l'albero.

```
' Inseriamo Nodi nell'albero
' Docenti
Set TmpNode = Me.R_AlberoTest.Nodes.Add(, , "Prof1", "Mario Guzzo", "Gruppo", "Gruppo")
TmpNode.Expanded = True
Set TmpNode = Me.R_AlberoTest.Nodes.Add(, , "Prof2", "Giovanni Pollini", "Gruppo", "Gruppo")
TmpNode.Expanded = True
Set TmpNode = Me.R_AlberoTest.Nodes.Add(, , "Prof3", "Alessia Gambacciani", "Gruppo", "Gruppo")
TmpNode.Expanded = True

' Studenti
Set TmpNode = Me.R_AlberoTest.Nodes.Add("Prof1", tvwChild, "Stud0", "Giorgio Virgo", "Utente", "Utente")
Set TmpNode = Me.R_AlberoTest.Nodes.Add("Prof2", tvwChild, "Stud1", "Monica Righi", "Utente", "Utente")
Set TmpNode = Me.R_AlberoTest.Nodes.Add("Prof3", tvwChild, "Stud2", "Alessandro Delbue", "Utente", "Utente")
Set TmpNode = Me.R_AlberoTest.Nodes.Add("Prof1", tvwChild, "Stud3", "Ascanio Verdi", "Utente", "Utente")
Set TmpNode = Me.R_AlberoTest.Nodes.Add("Prof2", tvwChild, "Stud4", "Yuri Pazzaglia", "Utente", "Utente")
Set TmpNode = Me.R_AlberoTest.Nodes.Add("Prof3", tvwChild, "Stud5", "Vera Gentili", "Utente", "Utente")
Set TmpNode = Me.R_AlberoTest.Nodes.Add("Prof1", tvwChild, "Stud6", "Sonia Frati", "Utente", "Utente")
Set TmpNode = Me.R_AlberoTest.Nodes.Add("Prof2", tvwChild, "Stud7", "Giorgio Esposito", "Utente", "Utente")
Set  TmpNode  =  Me.R_AlberoTest.Nodes.Add("Prof3",  tvwChild,  "Stud8",  "Riccardo  Funghetti",  "Utente",
"Utente")
       Set  TmpNode  =  Me.R_AlberoTest.Nodes.Add("Prof1",  tvwChild,  "Stud9",  "Sergio  Sgabelli  Vinti",
"Utente", "Utente")
```

Il risultato del precedente codice è mostrato nella seguente figura:



Il passo successivo è quello di abilitare la funzionalità Drag – and – Drop (D&D). Selezionare, dunque, il controllo TreeView **R_AlberoTest** nella maschera **ProvaDragDrop** e col tasto destro del mouse far apparire la maschera delle proprietà del controllo (vedi figura seguente).

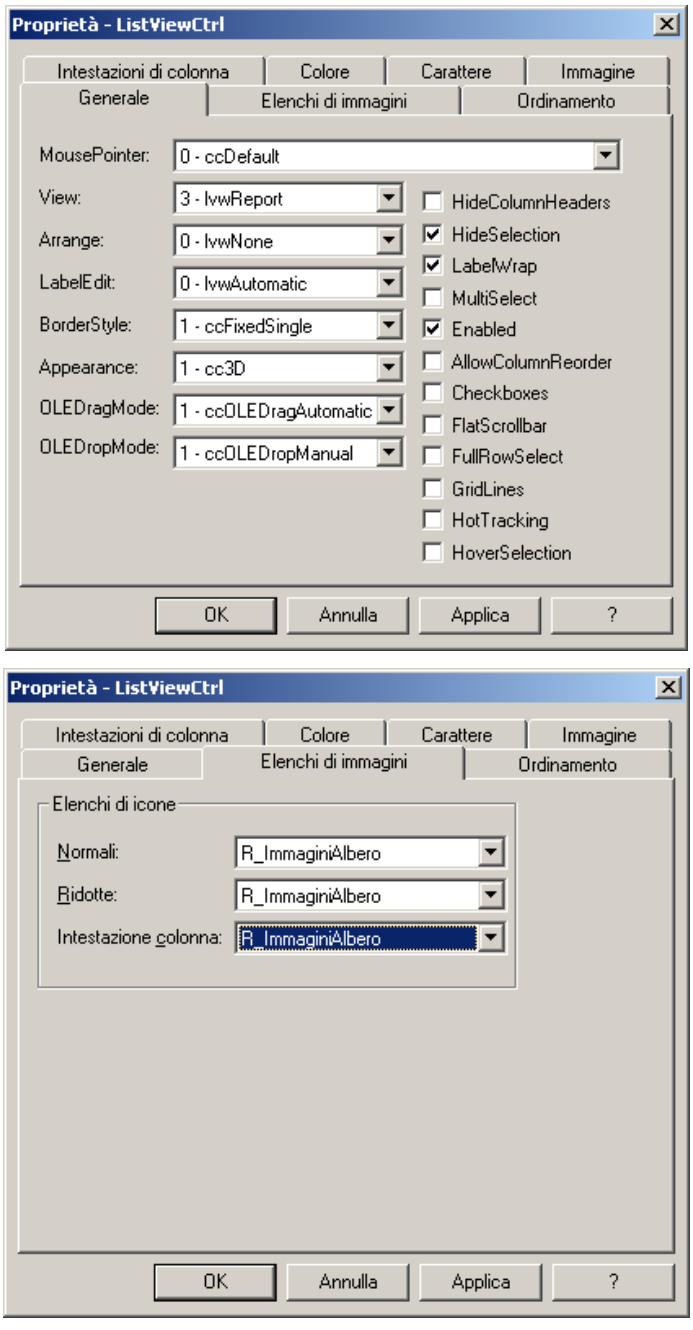


Impostare, quindi, le seguenti proprietà come di seguito indicato:

- OLEDragMode: 1- ccOLEDragAutomatic;

- OLEDropMode: 1 – ccOLEDropManual.

Inseriamo nella maschera anche un controllo ListView, denominato R_ListaTest, ed impostiamone proprietà in modo da abilitarne la modalità *Drag and Drop* ed associamogli le immagini del conrollo ImageList. Per ulteriori dettagli sul controllo ActiveX *List View* vedere [3].



Riempiamo, quindi, il nuovo controllo con una lista di oggetti (studenti), sempre sfruttando l'evento ***Form_Load()***, come già fatto per il controllo *TreeView*.

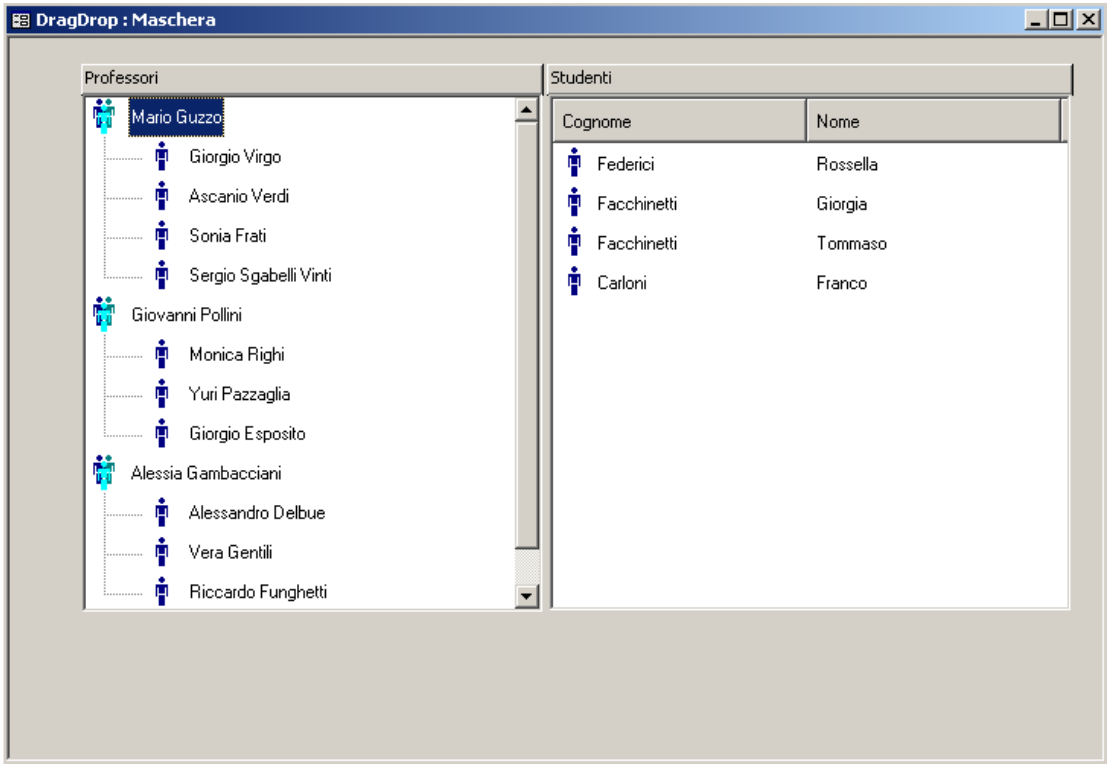
Il codice del popolamento dei due predetti controlli nella procedura *Form_Load* è il seguente:

```
Private Sub Form_Load()
Dim TmpNode As MSComctlLib.Node
Dim TmpItem As MSComctlLib.ListItem
Dim CH As MSComctlLib.ColumnHeader

' Inseriamo Nodi nell'albero
' Docenti
Set TmpNode = Me.R_AlberoTest.Nodes.Add(, , "Prof1", "Mario Guzzo", "Gruppo", "Gruppo")
TmpNode.Expanded = True
Set TmpNode = Me.R_AlberoTest.Nodes.Add(, , "Prof2", "Giovanni Pollini", "Gruppo", "Gruppo")
TmpNode.Expanded = True
Set TmpNode = Me.R_AlberoTest.Nodes.Add(, , "Prof3", "Alessia Gambacciani", "Gruppo", "Gruppo")
TmpNode.Expanded = True

' Studenti
Set TmpNode = Me.R_AlberoTest.Nodes.Add("Prof1", tvwChild, "Stud0", "Giorgio Virgo", "Utente",
"Utente")
Set TmpNode = Me.R_AlberoTest.Nodes.Add("Prof2", tvwChild, "Stud1", "Monica Righi", "Utente",
"Utente")
Set TmpNode = Me.R_AlberoTest.Nodes.Add("Prof3", tvwChild, "Stud2", "Alessandro Delbue", "Utente",
"Utente")
Set TmpNode = Me.R_AlberoTest.Nodes.Add("Prof1", tvwChild, "Stud3", "Ascanio Verdi", "Utente",
"Utente")
Set TmpNode = Me.R_AlberoTest.Nodes.Add("Prof2", tvwChild, "Stud4", "Yuri Pazzaglia", "Utente",
"Utente")
Set TmpNode = Me.R_AlberoTest.Nodes.Add("Prof3", tvwChild, "Stud5", "Vera Gentili", "Utente",
"Utente")
Set TmpNode = Me.R_AlberoTest.Nodes.Add("Prof1", tvwChild, "Stud6", "Sonia Frati", "Utente", "Utente")
Set TmpNode = Me.R_AlberoTest.Nodes.Add("Prof2", tvwChild, "Stud7", "Giorgio Esposito", "Utente",
"Utente")
Set TmpNode = Me.R_AlberoTest.Nodes.Add("Prof3", tvwChild, "Stud8", "Riccardo Funghetti", "Utente",
"Utente")
Set TmpNode = Me.R_AlberoTest.Nodes.Add("Prof1", tvwChild, "Stud9", "Sergio Sgabelli Vinti", "Utente",
"Utente")

' Riempimento di studenti nel controllo ListView
Me.R_ListaTest.ColumnHeaders.Clear
' Impostazione delle colonne
Set CH = Me.R_ListaTest.ColumnHeaders.Add(1, "Cognome", "Cognome", 2500, lvwColumnLeft)
Set CH = Me.R_ListaTest.ColumnHeaders.Add(2, "Nome", "Nome", 2500, lvwColumnLeft)
' Inserimento di quattro studenti
Me.R_ListaTest.ListItems.Clear
Set TmpItem = Me.R_ListaTest.ListItems.Add(1, "Stud10", "Carloni", "Utente", "Utente")
TmpItem.SubItems(1) = "Franco"
Set TmpItem = Me.R_ListaTest.ListItems.Add(1, "Stud11", "Facchinetti", "Utente", "Utente")
TmpItem.SubItems(1) = "Tommaso"
Set TmpItem = Me.R_ListaTest.ListItems.Add(1, "Stud12", "Facchinetti", "Utente", "Utente")
TmpItem.SubItems(1) = "Giorgia"
Set TmpItem = Me.R_ListaTest.ListItems.Add(1, "Stud13", "Federici", "Utente", "Utente")
TmpItem.SubItems(1) = "Rossella"
Me.R_ListaTest.Requery
```

```
End Sub
```

.

La Form appare, quindi, in questo modo:



Se si prova a selezionare uno studente in uno dei controlli e, premendo il tasto sinistro del mouse, si prova a trascinarlo all'interno del controllo stesso o nell'altro controllo, il puntatore cambierà forma ad indicare che è sono abilitate le funzionalità D&D.

Gli eventi di gestione della funzionalità Drag – and –Drop (D&D)

Una volta attivata le funzionalità D&D, per controllarne il funzionamento a tempo di esecuzione è possibile impiegare i seguenti eventi:

1. *_OLEStartDrag*(): invocato all'inizio del trascinamento;
2. *_OLEDragOver()*: invocato quando un oggetto è trascinato sopra il controllo;
3. *_OLEDragDrop()*: invocato al termine del trascinamento, quando si rilascia il tasto sinistro del mouse;

Il controllo, inoltre, fornisce un metodo ed una proprietà di grande utilità per la funzionalità Drag – and - Drop: Il metodo ***HitTest()*** e la proprietà ***DropHighlight()***.

Nel seguito illustreremo le funzionalità Drag-and-Drop prima nel caso di trascinamento di un Nodo (*Node*) da un punto di un controllo TreeView ad un altro punto del medesimo controllo, poi nel caso di trascinamento di un oggetto (*Item*) da un controllo ListView ad un punto di un controllo TreeView.

OleStartDrag()

Questo evento viene invocato prima che inizi l'azione di trascinamento del nodo selezionato e viene impiegato, solitamente, per operazioni di *controllo* e/o di *inizializzazione*. La sintassi completa dell'evento è la seguente:

Oggetto_OleStartDrag(Data As Object, AllowedEffect as Long)

Oggetto	È l'oggetto da cui ci sia aspetta l'invocazione dell'evento, nel caso in esempio è il controllo **R_AlberoTest**.
Data	E' un oggetto contenente I formati che il controllo fornirà ed, eventualmente, i dati per quei formati.
AllowedEffect	E' un intero lungo che specifica gli effetti che il controllo supporta. I valori possibili sono i seguenti: (0) – **vbDropEffectNone**: L'elemento di destinazione dell'operazione di trascinamento (Drag) non può accettare dati; (1) – **vbDropEffectCopy**: L'esecuzione dell'operazione di Drop si conclude con una **copia** dell'elemento sorgente trascinato sull'elemento di destinazione, lasciando comunque l'elemento sorgente inalterato; (2) – **vbDropEffectMove**: L'esecuzione dell'operazione di Drop si conclude con uno **spostamento** dell'elemento sorgente trascinato sull'elemento di destinazione.

Esempio:

Come regola generale di consistenza logica dei dati, dobbiamo imporre il vincolo che un professore non possa essere assegnato ad un altro professore o ad uno studente. Questa regola la possiamo implementare impedendo le operazioni di D&D sui nodi di primo livello dell'albero, andando a controllare il tipo di nodo che si sta per trascinare catturando l'evento **OleStartDrag**.

Catturiamo, dunque, l'evento **R_AlberoTest_OleStartDrag** come nel seguente frammento di codice:

```
Private Sub R_AlberoTest_OleStartDrag(Data As Object, AllowedEffect As Long)
' Verifica di selezione di un nodo di secondo livello: non si possono spostare
' i professori, cioè i nodi di primo livello
Dim ND As MSComctlLib.Node
Set ND = Me.R_AlberoTest.SelectedItem
If Not ND Is Nothing Then
    If ND.Parent Is Nothing Then
        MsgBox "Non si può spostare un professore.", , "Professori"
    End If
End If
End Sub
```

La procedura che abbiamo definito nel precedente frammento di codice mostrerà il seguente messaggio se si prova a trascinare un nodo di primo livello:.

Osservazione:

Si consideri la riga:

Set ND=Me.R_AlberoTest.SelectedItem

Non è detto che il Nodo che si sta iniziando a trascinare sia effettivamente quello selezionato. Per essere sicuri che lo sia è necessario impostare da codice la proprietà **SelectedItem** catturando l'evento **MouseDown**.

Le sintassi esposti nell' *Object Browser* e nella documentazione in linea di Visual Basic per il metodo **MouseDown (MouseUp)** sono, rispettivamente:

```
Private Sub oggetto_MouseDown([index As Integer,]button As Integer, shift As Integer, x As
OLE_XPOS_PIXELS, y As OLE_YPOS_PIXELS)
```

Ovvero (Guida in linea VBA)

```
Private Sub oggetto_MouseDown([index As Integer,]button As Integer, shift As Integer, x As Single, y As Single)
```

Entrambe le sintassi generano errori di compilazione. La sintassi corretta da impiegare in VBA è la seguente:

```
Private Sub oggetto_MouseDown(ByVal Button As Integer, ByVal Shift As Integer, ByVal x As Long, ByVal y As Long)
```

Nella procedura che gestisce l'evento MouseDown, dunque, si deve impostare il nodo selezionato (*SelectedItem*) mediante il metodo **HitTest()** che illustreremo nel seguito del presente tutorial.

```
Private Sub R_AlberoTest_MouseDown(ByVal Button As Integer, ByVal Shift As Integer, ByVal x As Long, ByVal y As Long)
' Si deve selezionare il nodo che si inizia a trascinare
'(non è sempre scontato che lo sia)!
Dim TmpNode As MSComctlLib.Node
Set Me.R_AlberoTest.SelectedItem = Me.R_AlberoTest.HitTest(x, y)
End Sub
```

OleDragOver()

Questo evento viene invocato durante l'operazione di trascinamento dell'elemento (Node/Item) selezionato sugli altri elementi del controllo ActiveX. Questo evento è utile per controllare la posizione del puntatore del mouse all'interno del controllo stesso.

La sintassi completa dell'evento è la seguente:

Oggetto_OleDragOver (*data *As Object,* effect *As Long,* button *As Integer,* shift *As Integer,* x *As Single,* y *As Single,* state *As Integer)

Oggetto	È' l'oggetto da cui ci sia aspetta l'invocazione dell'evento, nel caso in esempio è il controllo **R_AlberoTest**.
Data	E' un oggetto contenente I formati che il controllo fornirà ed, eventualmente, i dati per quei formati.
effect	Un intero lungo (Long) inizializzato dall'oggetto origine per identificare tutti gli *effetti* supportati.
button	Valore intero che agisce come un campo di bit corrispondente allo stato di uno dei tasti

	del mouse quando è premuto. Il tasto sinistro corrisponde al bit in posizione 0, il tasto destro corrisponde al bit in posizione 1 ed il tasto centrale corrisponde al bit in posizione 2. Per ciascun bit il valore 1 indica che il tasto è premuto.
Shift	Valore intero che agisce come un campo di bit corrispondenti allo stato dei tasti *Shift*, *ctrl* e *alt* quando sono premuti. Il tasto *shift* corrisponde al bit 0, il tasto *ctrl* corrisponde al bit 1 ed il tasto *alt* corrisponde al bit 2. Per ciascun bit il valore 1 indica che il tasto è premuto
x, y	Sono valori interi che specificano l'attuale posizione del puntatore del mouse in coordinate orizzontali (x) e verticali (y).
State	E' un valore intero che corrisponde allo stato della transazione del Controllo in fase di trascinamento con riferimento alla maschera o al controllo di destinazione: (0) – Enter (il Controllo sorgente è trascinato all'interno della destinazione); (1) – Leave (il Controllo sorgente è trascinato all'esterno della destinazione); (2) – Over (il Controllo sorgente è stato spostato da una posizione ad un'altra all'interno della destinazione).

Esempio:

Con la seguente procedura è possibile evidenziare il nodo dell'albero sul quale il puntatore del mouse "*passa sopra*" durante l'operazione di trascinamento.

```
Private Sub R_AlberoTest_OLEDragOver(Data As Object, Effect As Long, _
Button As Integer, Shift As Integer, x As Single, y As Single, state As Integer)
' Evidenziazione del cursore
Me.R_AlberoTest.DropHighlight = Me.R_AlberoTest.HitTest(x, y)
End Sub
```

OleDragDrop()

Questo evento viene invocato al termine dell'operazione di trascinamento di un elemento di un controllo su un altro elemento. L'operazione di trascinamento ha termine quando l'elemento spostato viene posizionato per trascinamento (con il tasto sinistro del mouse premuto) su un altro elemento e viene rilasciato il tasto sinistro del mouse. Possiamo sfruttare questo evento per aggiornare lo stato del controllo ActiveX, per controllare la consistenza delle informazioni che si stanno spostando mediante trascinamento e per compiere le operazioni associate al rilascio (*Drop*) di un oggetto (*Item/Node*) su un altro oggetto (*Item/Node*).

La sintassi completa dell'evento è la seguente:

OLEDragDrop(*data *As Object,* Effect *As Long,* button *As Integer,* shift *As Integer,* x *As Single,* y *As Single,* state *As Integer)

Oggetto	È' l'oggetto da cui ci sia aspetta l'invocazione dell'evento, nel caso in esempio è il controllo **R_AlberoTest**.
Data	E' un oggetto contenente I formati che il controllo fornirà ed, eventualmente, i dati per quei formati.
Effect	Un intero lungo (Long) inizializzato dall'oggetto origine per identificare tutti gli *effetti* supportati.
Button	Valore intero che agisce come un campo di bit corrispondente allo stato di uno dei tasti del mouse quando è premuto. Il tasto sinistro corrisponde al bit in posizione 0, il tasto destro corrisponde al bit in posizione 1 ed il tasto centrale corrisponde al bit in posizione 2. Per ciascun bit il valore 1 indica che il tasto è premuto.
Shift	Valore intero che agisce come un campo di bit corrispondenti allo stato dei tasti *Shift*, *ctrl* e *alt* quando sono premuti. Il tasto *shift* corrisponde al bit 0, il tasto *ctrl* corrisponde al bit 1 ed il tasto *alt* corrisponde al bit 2. Per ciascun bit il valore 1 indica che il tasto è premuto
x, y	Sono valori interi che specificano l'attuale posizione del puntatore del mouse in coordinate orizzontali (x) e verticali (y).

Esempio:

Sfrutteremo l'evento OLEDragDrop per disegnare il nuovo assetto dell'albero ovvero:

1) rimuovere il nodo trascinato (Studente) dall'l'insieme di quelli associati al medesimo nodo di primo livello di origine (Professore);
2) Inserire il nodo trascinato (Studente) nell'insieme di quelli associati al nodo di primo livello (Professore) di destinazione, considerando anche che il nodo sul quale avviene materialmente l'operazione di rilascio (*Drop*) può essere sia un nodo di primo livello (Professore), sia un nodo di secondo livello (Studente): in quest'ultimo caso, per eseguire ugualmente e correttamente l'associazione, si deve risalire al corrispondente nodo di primo livello. In altre parole, se si rilascia uno Studente su un altro Studente, l'associazione viene effettuata col Professore a relativo a questo ultimo.
3) Ridisegnare il controllo ActiveX.

```
Private Sub R_AlberoTest_OLEDragDrop(Data As Object, Effect As Long, Button As Integer, shift As
Integer, x As Single, y As Single)
Dim Dest_Node As MSComctlLib.Node       'Nodo di destinazione
Dim Src_Node As MSComctlLib.Node        'Nodo di origine
Dim counter As Long                     'contatore

… omesso …

' prelievo del nodo d'origine
```

```
Set Src_Node = Me.R_AlberoTest.SelectedItem

' Prelievo del Nodo di Destinazione
Set Dest_Node = Me.R_AlberoTest.DropHighlight

' Se il Nodo di destinazione è di secondo livello, prelievo il relativo nodo
' di primo livello
If Dest_Node Is Nothing Then Exit Sub
If (Not Dest_Node.Parent Is Nothing) Then Set Dest_Node = Dest_Node.Parent

' rimozione del nodo di origine dall'albero
counter = 1
' ciclo di ricerca dell'indice del nodo di origine nella collezione Nodes
Do While counter <= Me.R_AlberoTest.Nodes.Count
    If Me.R_AlberoTest.Nodes.Item(counter) = Src_Node Then Exit Do
    counter = counter + 1
Loop
Me.R_AlberoTest.Nodes.Remove counter

' inserimento del nodo
Set Src_Node = Me.R_AlberoTest.Nodes.Add(Dest_Node.Key, tvwChild, Src_Node.Key, Src_Node.Text,
Src_Node.Image, Src_Node.SelectedImage)
' refresh
Me.R_AlberoTest.Requery
End Sub
```

La funzionalità Drag – and – Drop tra due controlli ActiveX

Abbiamo finora trattato il funzionamento delle funzionalità Drag – and - Drop (D&D) limitatamente al trascinamento di oggetti all'interno del medesimo controllo, in particolare abbiamo operativamente fatto riferimento ad un controllo di tipo *Tree View*. Quanto detto si può specularmente applicare ad un controllo di tipo *List View*.

In questo paragrafo affronteremo la problematica del trascinamento di un oggetto *Node* o *ListItem* da un controllo ad un altro all'interno della medesima maschera. In particolare vedremo operativamente come spostare le informazioni collezionate in un controllo di tipo *List View* per riorganizzarle in un controllo di tipo *TreeView* e ci serviremo ancora del precedente esempio di attribuzione delle entità **Studenti** alle entità **Professori**.

Quanto illustreremo è di validità generale, potendosi applicare con poche e semplici modifiche al trascinamento di informazioni da un controllo di tipo *Tree View* verso uno di tipo *List View*, oppure da un controllo di tipo *List View* verso un altro controllo di tipo *List View* o, ancora, tra due controlli di tipo *Tree View.*

Partiamo facendo due considerazioni:

1) il controllo *ListView* supporta la gestione degli eventi OLEDragStart, OLEDragOver ed OLEDragDrop in modo totalmente identico al controllo *TreeView*;
2) Nel trascinamento d'informazioni tra due controlli diversi abbiamo la necessità di identificare il controllo d'origine, cioè il controllo dal quale inizia il trascinamento (ovviamente questo problema non sussiste in operazioni D&D all'interno del medesimo controllo ActiveX).

Nella sintassi dei gestori degli eventi OLEDragStart, OLEDragOver ed OLEDragDrop compare, come primo parametro, un oggetto di Classe *DataObject*. L'oggetto *DataObject* può essere pensato come una specie di *clipboard*.

Attenzione: *se all'interno di un modulo VBA si invocano i predetti gestori con una firma in cui il primo parametro è dichiarato come oggetto di classe DataObject, si ottiene un errore. Per la sintassi che non genera errori nell'ambiente VBA, vedere gli esempi che seguono*.

Nella procedura che gestisce l'evento di inizio del trascinamento (*StartDrag*) nel controllo *Tree View*, scriviamo un'informazione identificativa del controllo stesso nell'oggetto di classe *DataObject* usando il metodo *SetData().*

Esempio:

```
Private Sub R_ListaTest_OLEStartDrag(Data As Object, AllowedEffect As Long)
' Salviamo l'informazione sul controllo d'origine
Data.SetData "R_ListaTest", 1    ' 1 = vbCfText indica che il tipo di informazioni portato dall'oggetto
                                 ' DataObject è una stringa di testo
End Sub
```

Nel codice d'esempio abbiamo impiegato il metodo *SetData()* per scrivere come stringa di testo (vbCfText=1) il nome del controllo *ListView* dal quale inizia l'operazione di trascinamento. Nella procedura che gestisce l'evento *DragDrop* degli altri controlli che possono accogliere l'informazione in corso di trascinamento si deve leggere dall'oggetto *DataObject* la stringa identificativa del controllo di partenza.

Esempio:

Il seguente frammento di codice, aggiunto alla procedura ***OLEDragDrop()*** del controllo *TreeView*, preleva l'informazione sul controllo d'origine, legge le informazioni necessarie alla creazione di un nuovo nodo (*Node*) dall'elemento *ListItem* selezionato, rimuove il predetto elemento *ListItem* del controllo *List View* ed esegue il refresh dei due controlli.

```
Private Sub R_AlberoTest_OLEDragDrop(Data As Object, Effect As Long, Button As Integer, shift As
Integer, x As Single, y As Single)
Dim Dest_Node As MSComctlLib.Node       'Nodo di destinazione
Dim Src_Node As MSComctlLib.Node        'Nodo di origine
Dim counter As Long                     'contatore
' Parte II
Dim Controllo As String                 'Stringa di appoggio
Dim Src_Item As MSComctlLib.ListItem    'ListItem d'Origine
Dim NomeIntero As String                'Stringa di Appoggio

' Verifica della provenienza del controllo
Controllo = Data.GetData(1)      ' 1 vbCFText

If Controllo = "R_ListaTest" Then
    ' PARTE DI CODICE NON OTTIMIZZATA PER RAGIONI DIDATTICHE
    ' Le informazioni provengono per trascinamento
    ' dal controllo ListView.

    ' Prelievo del Nodo di Destinazione
    If Me.R_AlberoTest.DropHighlight Is Nothing Then Exit Sub
    Set Dest_Node = Me.R_AlberoTest.DropHighlight
    ' Prelievo dell'Item d'origine
    Set Src_Item = Me.R_ListaTest.SelectedItem
    ' Composizione del Nome Intero
    NomeIntero = Src_Item.SubItems(1) & " " & Src_Item.Text
```

```
    ' Rimozione dell'Item dal controllo ListView
    Me.R_ListaTest.ListItems.Remove Src_Item.Key
    ' Se il Nodo di destinazione è di secondo livello, prelievo il relativo nodo
    ' di primo livello
    If (Not Dest_Node.Parent Is Nothing) Then Set Dest_Node = Dest_Node.Parent
    ' Aggiunta dell'Item al controllo TreeView
    Set Src_Node = Me.R_AlberoTest.Nodes.Add(Dest_Node.Key, tvwChild, Src_Item.Key, NomeIntero,
Src_Item.Icon, Src_Item.Icon)
    ' Refresh dei controlli
    Me.R_AlberoTest.Requery
    Me.R_ListaTest.Requery
    Exit Sub
End If
… contitnua …
…
```

Ulteriori considerazioni sulla funzionalità Drag-and-Drop

Tratteremo in questo paragrafo conclusivo la sintassi del metodo **HitTest()** e la proprietà ***DropHighLight*** e spiegheremo come superare uno dei maggiori limiti del controllo TreeView che si manifesta durante il trascinamento di un nodo.

Il metodo HitTest()

Questo metodo degli oggetti TreeView e ListView restituisce, rispettivamente, l'oggetto di classe Node e l'oggetto di classe ListItem corrispondente alle coordinate x, y del puntatore del mouse all'interno del controllo. Le coordinate del mouse vengono passate alla funzione come parametri.

Node TreeView_Object.HitTest(x as Single, y as Single);

ListItem ListView_Object.HitTest(x as Single, y as Single).

La proprietà DropHighLight

Questa proprietà degli oggetti *TreeView* e *ListView* consente di evidenziare un oggetto Node o ListItem. Con la riga di codice

```
' Evidenziazione del cursore
Me.R_AlberoTest.DropHighlight = Me.R_AlberoTest.HitTest(x, y)
```

Si evidenzia l'oggetto sopra il quale sta passando il puntatore del cursore durante le operazioni di trascinamento.

Abilitare lo scrolling durante il trascinamento di un nodo all'interno di un controllo TreeView

Uno dei più grandi limiti del controllo *Tree View*, così come documentato dall'articolo Q177743 del 13 Luglio 2004 della Microsoft Knowledge Base, è il fatto che durante l'operazione di trascinamento oltre i confini del controllo, il controllo stesso non esegue automaticamente lo scrollino verticale.

Per superare questo limite si deve ricorrere necessariamente alle API di Windows (libreria user32.dll) mandando un messaggio al controllo *Tree View* in modo che effettui lo scrolling verticale. La funzione delle API di windows da invocare per inviare un messaggio al

controllo *Tree View* è ***SendMessageA***, la cui dichiarazione all'interno di un modulo VBA è del tipo:

```
Private Declare Function sendMessage Lib "user32" Alias "SendMessageA" (ByVal hwnd As Long, ByVal wMsg
As Long, ByVal wParam As Long, lParam As Any) As Long
```

Abbiamo ritenuto conveniente invocare la funzione all'interno di una procedura nella quale sia possibile ottenere informazioni sulla posizione del puntatore le mouse.

Esempio:

Nell'esempio esaminato nel presente documento, abbiamo invocato la funzione ***sendMessage()*** all'interno della procedura che gestisce l'evento ***DragOver***:

```
Private Sub R_AlberoTest_OLEDragOver(Data As Object, Effect As Long, Button As Integer, shift As
Integer, x As Single, y As Single, state As Integer)
' Evidenziazione del cursore
Me.R_AlberoTest.DropHighlight = Me.R_AlberoTest.HitTest(x, y)
If y < 10 Then
    ' scroll verso l'alto
    sendMessage Me.R_AlberoTest.hwnd, 277&, 0&, vbNull
End If
If y > Me.R_AlberoTest.Height - 100 Then
    ' scroll verso il basso
    sendMessage Me.R_AlberoTest.hwnd, 277&, 1&, vbNull
End If
End Sub
```

Come si può osservare, la funzione di sistema viene invocata passandogli un *handler* al controllo *TreeView*, il codice del messaggio (277&) e la direzione in cui effettuare lo scrolling (0& verso l'alto, 1&verso il basso) quando la coordinata verticale (y) supera due soglie: una inferiore ed una superiore.

Conclusioni

Questo documento conclude una serie di quattro tutorial dedicati ai controlli ActiveX *List View*, *Tree View* e *ImageList*. Lo scopo che speriamo di aver raggiunto con questi documenti non è quello di aver fornito una trattazione completa ed esaustiva dei predetti controlli ma quello di aver raccolto un insieme informazioni, spesso esigue, disomogenee e sparpagliate, ed esperienze di programmazione allo scopo di consentire l'impiego questi controlli in modo rapido ed efficace.

Riferimenti

[1] Tutorial – ActiveX – Controllo Tree View V.1 del 30/08/2002;

[2] Tutorial – ActiveX – Oggetto ImageList: Aggiungere immagini ad un controllo Common Control V.1.del 12/12/2004;

[3] Tutorial – ActiveX – Controllo List View, V1 del 21/12/2004.