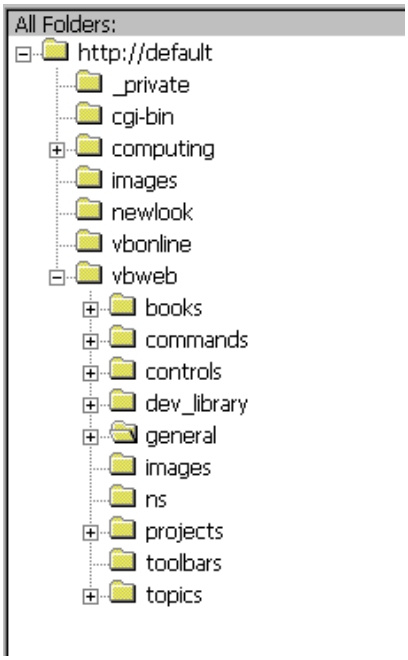
Contents

TreeView Control - Introduction

Author James Crowley

Introduction

The Tree View control is a Visual Basic version of the control you see used in many programs, including Explorer and FrontPage, used to list the folders on your hard disk. This control allows you to add nodes to a tree, each of which can have sub items. Below is an image of the Tree View control in use.



The Tree View control comes with all editions of Visual Basic, except the standard version.

To add a list view control to your project, goto Project|Components. Now scroll down to Microsoft Common Controls and check the box next to it. Click OK. You will now see a number of new controls added to your toolbox. The TreeView icon looks like this:

Adding Nodes

Adding an item, or Node to the TreeView control is simple, once you know how. Adding a node uses the following syntax:

```
TreeView1.Nodes.Add Relative, Relationship, Key, Text, Image, SelectedImage
```

All of this parameters are optional. As the TreeView control is made up of items in a Tree, Visual Basic needs to know where the new item goes. You use Relative and Relationship to specify this. Relative specifies a relation (an existing node) to the new node you are creating. Relationship specifies how the new node is related to this existing node. Possible relationships are:

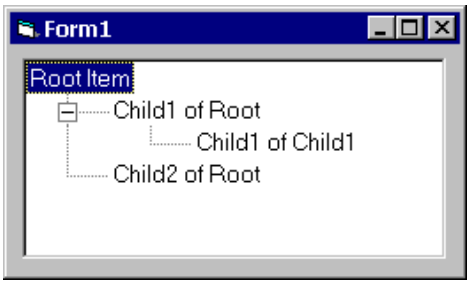
tvwChild '// new node is a child (one level down) of the specified node
tvwFirst '// new node is first, at the same level of the specified node
tvwLast '// new node is last, at the same level of the specified node
tvwNext '// new node is after (at the same level) the specified node

tvwPrevious '// new node is before (at the same level) the specified node

Key is a unique indentifier for this node (but easier to remember than an Index ID), which can be a string. Text is the text to be displayed for this node. Image is the image to be displayed next to the node, specified from the bound Image List (discussed later), such as a folder. SelectedImage is the image to be displayed when the node is selected, such as an open folder, specified from the bound Image list (discussed later). Add a TreeView control to your form and enter the code below.

```
Private Sub Form_Load()
    '// add some items
    TreeView1.Nodes.Add , , "root", "Root Item"
    TreeView1.Nodes.Add "root", tvwChild, "child1_root", "Child1 of Root"
    TreeView1.Nodes.Add "root", tvwChild, "child2_root", "Child2 of Root"
    TreeView1.Nodes.Add "child1_root", tvwChild, "child1_child1", "Child1 of Child1"
    '// make sure Child1 of Child1 is visible
    TreeView1.Nodes("child1_child1").EnsureVisible
End Sub
```

Run the project. You should see something like the image below:



Notice that the last line in the Form_Load procedure uses the EnsureVisible method. This forces the specified node to become visible, by expanding another item in needed.

Removing Nodes

To remove an node, you either need to know its Index or its unique key (if you gave it one). Then you use the following syntax:

```
TreeView1.Nodes.Remove IndexOrKey
```

The following example first removes a node which key is "root", and then removes a node which key is 12.

```
TreeView1.Nodes.Remove "root"
TreeView1.Nodes.Remove 12
```

Note that if a node has sub items when it is removed, all its sub items are removed too.

If you want to remove all the nodes in the TreeView, you can use the Clear method:

```
TreeView1.Nodes.Clear
```

TreeView Styles

The TreeView control has a number of properties that changes the way it looks. A few are discussed below.

Indentation - Sets the amount distance between a node and its children.

BorderStyle - Sets the style of the border. Either ccNone (no border) or ccFixedSingle (single border)

Appearance - Sets the appearance of the TreeView control. Either cc3D (3D style appearance) or ccFlat (flat appearance)

LineStyle - Sets the line style. Either tvwRootLines (lines come from the very top) or tvwTreeLines (lines come from each item at the top level). The best way to see this is to try it at home!

Style - Sets the treeview style. Can be set to one of the following:

tvwTextOnly - Only text is displayed. No treelines, images or plus/minus's (expand/shrink) are displayed
tvwPictureText - Text and pictures are displayed. No treelines or plus/minus's.
tvwPlusMinusText - Text and plus/minus's are displayed. No treelines or images.
tvwPlusPictureText - Text, plus/minus's and images are displayed. No treelines.
tvwTreelinesText - Text and treelines are displayed. No images or plus/minus's.
tvwTreelinesPictureText - Text, images and treelines are displayed. No plus/minus's
tvwTreelinesPlusMinusText - Text, treelines and plus/minus's are displayed. No images
tvwTreelinesPlusMinusPictureText - Text, treelines, plus/minus's and images are all displayed.

Please note that you need to assign an image to each item for an image to be displayed.

Using Images

Like the listview control, the treeview control needs to be 'bound' to an image control. To do this, add an ImageList Control to your form, name it, and add any images you want to use in the TreeView control to it. Be sure to set the image size first (normally set to 16x16). Set keys if you wish, as this makes it easier to identify each image. It is best to stick with one case system (ie all caps, all lowercase etc).

Now, change the ImageList property of the TreeView control to the name of the ImageList you want to use. Now, when you add a new node to the control, you can specify and image index or key too. That image will then be displayed next to the node. The following code adds a new node, with the image "TextFile" displayed next to it.

```
TreeView1.Nodes.Add , , , "TestImageItem", "TextFile"
```

You can also specify a image to be displayed when the item is selected . The following code adds a new node. The image "FolderClosed" will be displayed, and when it is selected, "FolderOpen" will be displayed (like the folders in explorer).

```
TreeView1.Nodes.Add , , , "TestFolderItem", "FolderClosed", "FolderOpen"
```

Note that in place of the text used to identify the image, you can also use its index:

```
TreeView1.Nodes.Add , , , "TestImageItem", 1
```

Editing Labels

The easiest way for the users to edit the labels (the text for each node) during run time is to set the TreeView control's LabelEdit property to tvwAutomatic. Now, like in explorer, when the item is clicked once, you can edit its caption

By using the AfterLabelEdit event, you can check that the new caption is valid, and that it is not empty:

```
Private Sub TreeView1_AfterLabelEdit(Cancel As Integer, NewString As String)
    If NewString = Empty Then
        '// new string is empty
        Msgbox "Please enter a new name"
        '// cancel edit
        Cancel = -1
    Else
        '// string ok
    End If
End Sub
```

There is also an BeforeLabelEdit event that occurs before the item is edited. You can use this to prevent the root item being edited, for example:

```
Private Sub TreeView1_BeforeLabelEdit(Cancel As Integer)
    If TreeView1.SelectedItem.Key = "root" Then
        '// item is root item
        Msgbox "You cannot edit the root items name"
        '// cancel edit
        Cancel = -1
    End If
End Sub
```

Expanding Items

If you want to make sure a node is visible, you can use the EnsureVisible method:

```
TreeView1.Nodes(21).EnsureVisible
```

This will expand any items, and scroll as required to ensure that the item specified is visible.

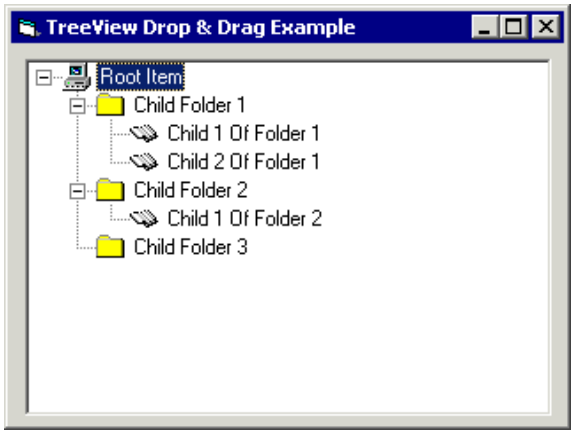
If a node has child items, a + appears next to it. The user can press this to show the child items. When the user expands an node, the Expand event occurs. Once the node has been expanded, a - appears next to it. When this is pressed, the child items are hidden, and the Collapse event occurs

Drag and Drop

This one look me a while to work out, but here it is.

Firstly, add a TreeView control to your project and name it tvProject. Next, add an Image List Control, set the image size to 16x16, add a 3 images, and set their keys to Root, FolderClosed and Item.

Then, set the TreeView's OLEDragMode to 1-ccOLEDragAutomatic, and its OLEDropMode to 1-ccOLEDropManual. Also, set its ImageList property to the ImageList control you have just added.



The following code will allow you to move nodes and its children around on the control (like in explorer when you drag folders to new locations etc).

```
'// variable that tells us if
'// we are dragging (ie the user is dragging a node from this treeview control
'// or not (ie the user is trying to drag an object from another
'// control and/or program)
Private blnDragging As Boolean

Private Sub Form_Load()
    '// fill the control with some dummy nodes
    With tvProject.Nodes
        .Add , , "Root", "Root Item", "Root"
        '// add some child folders
        .Add "Root", tvwChild, "ChildFolder1", "Child Folder 1", "FolderClosed"
        .Add "Root", tvwChild, "ChildFolder2", "Child Folder 2", "FolderClosed"
        .Add "Root", tvwChild, "ChildFolder3", "Child Folder 3", "FolderClosed"
        '// add some children to the folders
        .Add "ChildFolder1", tvwChild, "Child1OfFolder1", "Child 1 Of Folder 1", "Item"
        .Add "ChildFolder1", tvwChild, "Child2OfFolder1", "Child 2 Of Folder 1", "Item"
        .Add "ChildFolder2", tvwChild, "Child1OfFolder2", "Child 1 Of Folder 2", "Item"
    End With
End Sub

Private Sub tvProject_MouseDown(Button As Integer, Shift As Integer, x As Single, y As Single)
    Dim nodNode As Node
    '// get the node we are over
```

```
    Set nodNode = tvProject.HitTest(x, y)
    If nodNode Is Nothing Then Exit Sub '// no node
    '// ensure node is actually selected, just incase we start dragging.
    nodNode.Selected = True
End Sub

'// occurs when the user starts dragging
'// this is where you assign the effect and the data.
Private Sub tvProject_OLEStartDrag(Data As MSComctlLib.DataObject, AllowedEffects As Long)
    '// Set the effect to move
    AllowedEffects = vbDropEffectMove
    '// Assign the selected item's key to the DataObject
    Data.SetData tvProject.SelectedItem.Key
    '// we are dragging from this control
    blnDragging = True
End Sub

'// occurs when the object is dragged over the control.
'// this is where you check to see if the mouse is over
'// a valid drop object
Private Sub tvProject_OLEDragOver(Data As MSComctlLib.DataObject, Effect As Long, Button As Integer, Shift
As Integer, x As Single, y As Single, State As Integer)
    Dim nodNode As Node
    '// set the effect
    Effect = vbDropEffectMove
    '// get the node that the object is being dragged over
    Set nodNode = tvProject.HitTest(x, y)
    If nodNode Is Nothing Or blnDragging = False Then
        '// the dragged object is not over a node, invalid drop target
        '// or the object is not from this control.
        Effect = vbDropEffectNone
    End If
End Sub

'// occurs when the user drops the object
'// this is where you move the node and its children.
'// this will not occur if Effect = vbDropEffectNone
Private Sub tvProject_OLEDragDrop(Data As MSComctlLib.DataObject, Effect As Long, Button As Integer, Shift
As Integer, x As Single, y As Single)
    Dim strSourceKey As String
    Dim nodTarget    As Node
    '// get the carried data
    strSourceKey = Data.GetData(vbCFText)
    '// get the target node
    Set nodTarget = tvProject.HitTest(x, y)
    '// if the target node is not a folder or the root item
    '// then get it's parent (that is a folder or the root item)
    If nodTarget.Image <> "FolderClosed" And nodTarget.Key <> "Root" Then
        Set nodTarget = nodTarget.Parent
    End If
    '// move the source node to the target node
    Set tvProject.Nodes(strSourceKey).Parent = nodTarge
    '// NOTE: You will also need to update the key to reflect the changes
    '// if you are using it
    '// we are not dragging from this control any more
    blnDragging = False
    '// cancel effect so that VB doesn't muck up your transfer
    Effect = 0
End Sub
```

James first started writing tutorials on Visual Basic in 1999 whilst starting this website (then known as VB Web). Since then, the site has grown rapidly, and James has written numerous tutorials, articles and reviews on VB, PHP, ASP and C#. In October 2003, James formed the company Developer Fusion Ltd, which owns this website, and also offers various development services. In his spare time, he's a 3rd year undergraduate studying Computer Science in the UK. He's also a Visual Basic MVP.

Related Content

- Dragging TreeView nodes
- Treeview Files/Folder Lists Recursively