Lab 5 – User Interface Design: List boxes and Trees

Human Computer Interfacing

(BESE 14)

Objectives

This is the 4th lab activity in designing a user interface using vb.Net. Today we will learn the use of the following controls:

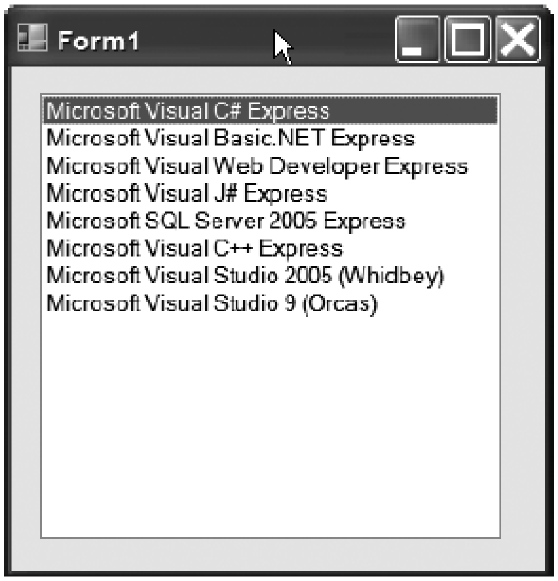
- ListBox
- TreeView

In addition, we will briefly see the vb.net Events.

Task 1 (Practice Exercises)

The ListBox Control

- As the name suggests, the ListBox control displays a simple list of items within a box and is widely used in many Windows applications.



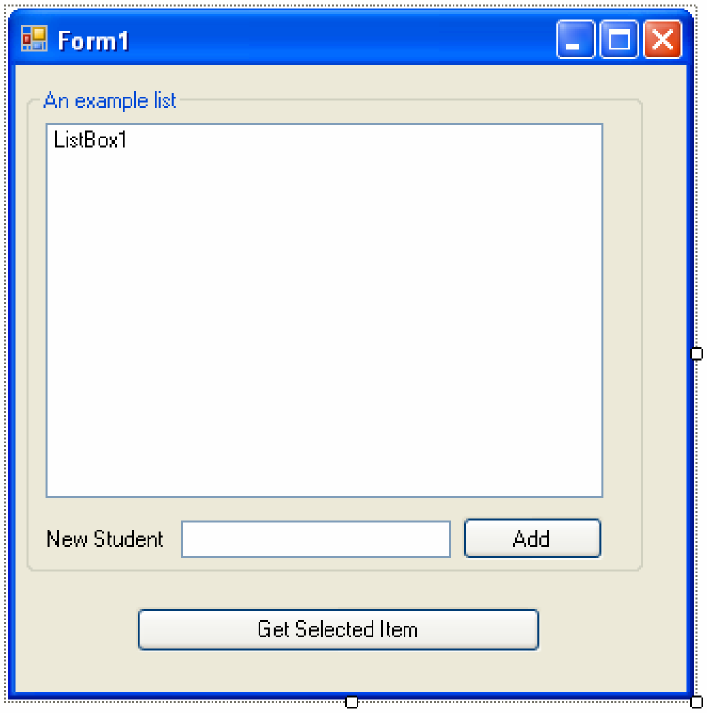
Microsoft Visual C# Express
Microsoft Visual Basic.NET Express
Microsoft Visual Web Developer Express
Microsoft Visual J# Express
Microsoft SQL Server 2005 Express
Microsoft Visual C++ Express
Microsoft Visual Studio 2005 (Whidbey)
Microsoft Visual Studio 9 (Orcas)

Figure 1 A simple ListBox Control

- The ListBox control exposes a property called Items that you can use to add items to the list. You can use this property either through code, or even in the

Properties window. In the form editor, just drop a ListBox on a form, and then find the “Items” property and click on the button (“…”) in the property’s input area to open the Editor.

- To use the ListBox Control, you will design a simple form (some thing like the one below) where the user could enter a String (e.g. Name of a student) in the TextBox and clicking on the Add Button, the Name would be added to the list.
- Once the list has been created, the user may select an Item and clicking on the second button (Get selected Item), a MessageBox will display the selected string.



- You could add some thing like this to the Add button:

```
ListBox1.Items.Add(TextBox1.Text)

TextBox1.Text = ""
```

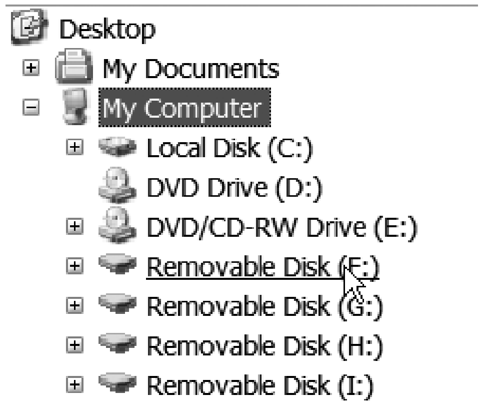
- The Items property on the ListBox is something called a Collection. Collections have an Add() method on them to add new items into the collection. In the case of ListBox.Items, this translates to adding an item into the list box. So, the code just takes the string of text entered into the text box and adds it into the Items collection. When that's done, the text box is cleared out, ready for the next piece of text to be entered.

- To get the selected item from the list you may use:

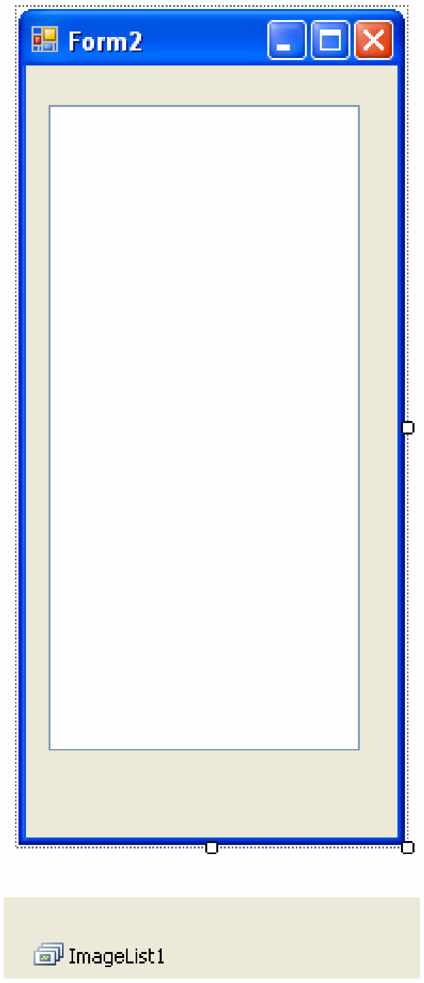
```
Dim selected As String = ListBox1.SelectedItem
```

The TreeView Control

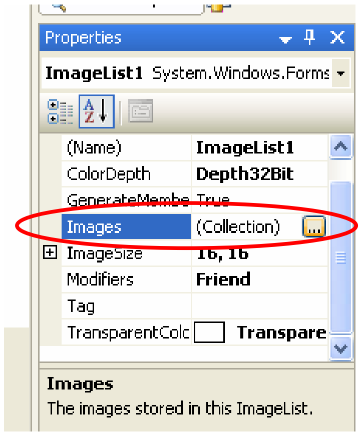
- The TreeView control is a unique list control, in that it displays a hierarchical view of data. You can find the TreeView Control in Windows Explorer:



- The TreeView holds a Nodes collection, each item in which is a TreeNode object. Each Node itself may contain a Collection of Nodes.
- You can change the actual view of the nodes by associating graphics with them.
- Create a Form and add the TreeView Control to it.
- Double-click the ImageList control in the Toolbox to drop it onto the form. The ImageList control is a nonvisual control, so it appears underneath the form. We will use this control to hold the images which will be visible in the Tree.
- You will have on your Form, some thing like this:



- Select the ImageList and Control and in the Properties Window, Go to the Property: Images



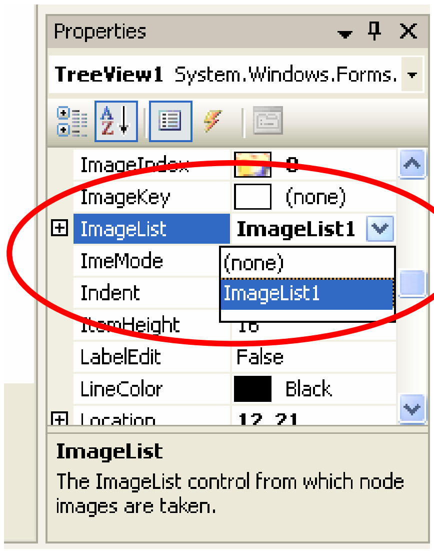
ImageList1 System.Windows.Forms

(Name)	ImageList1
ColorDepth	Depth32Bit
GenerateMembe	True
Images	(Collection)
ImageSize	16, 16
Modifiers	Friend
Tag	
TransparentColc	Transpare

Images
The images stored in this ImageList.

- Add few Images to this Control and return back to the Form.

- Select the TreeView, find the property ImageList and select ImageList1. We are basically telling the Tree control to use images from ImageList1.



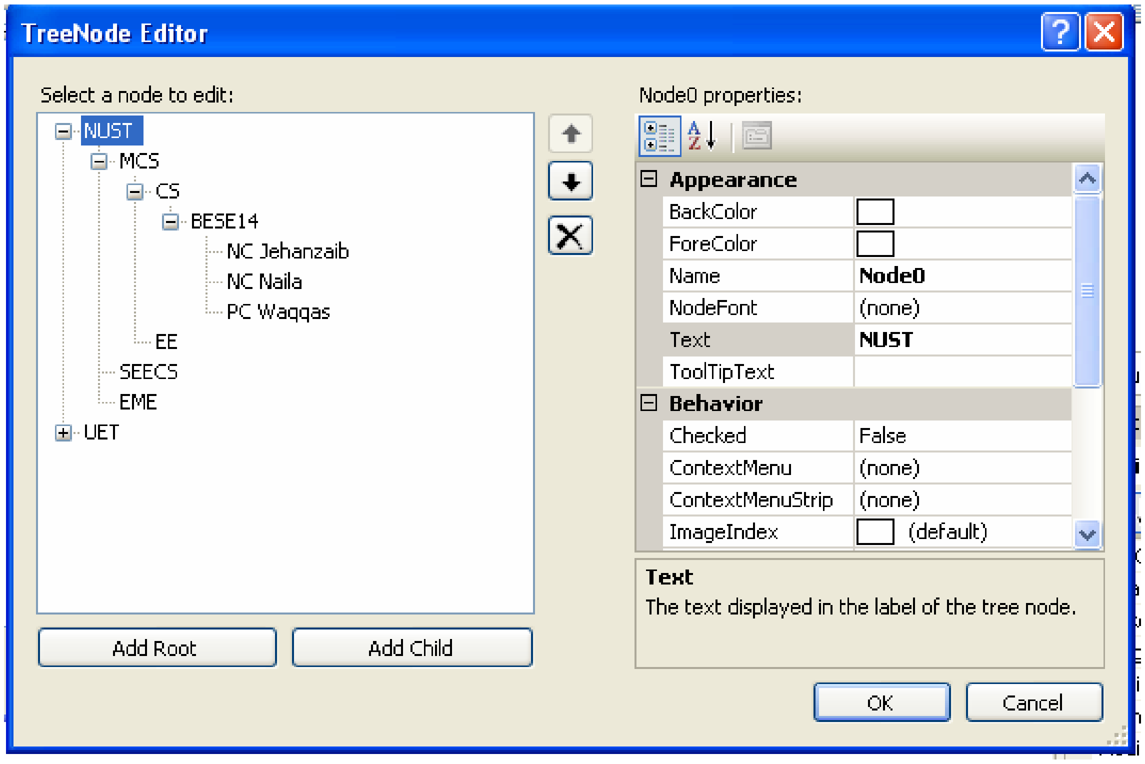
Properties

TreeView1 System.Windows.Forms.

ImageIndex	0
ImageKey	(none)
ImageList	ImageList1
ImeMode	(none)
Indent	ImageList1
ItemHeight	16
LabelEdit	False
LineColor	Black
Location	12, 21

ImageList
The ImageList control from which node images are taken.

- With the TreeView Control selected, Click the Nodes Property to add Nodes to the Tree.
- Using the Add Root and Add Child buttons, you can create a Tree like this:



TreeNode Editor

Select a node to edit:

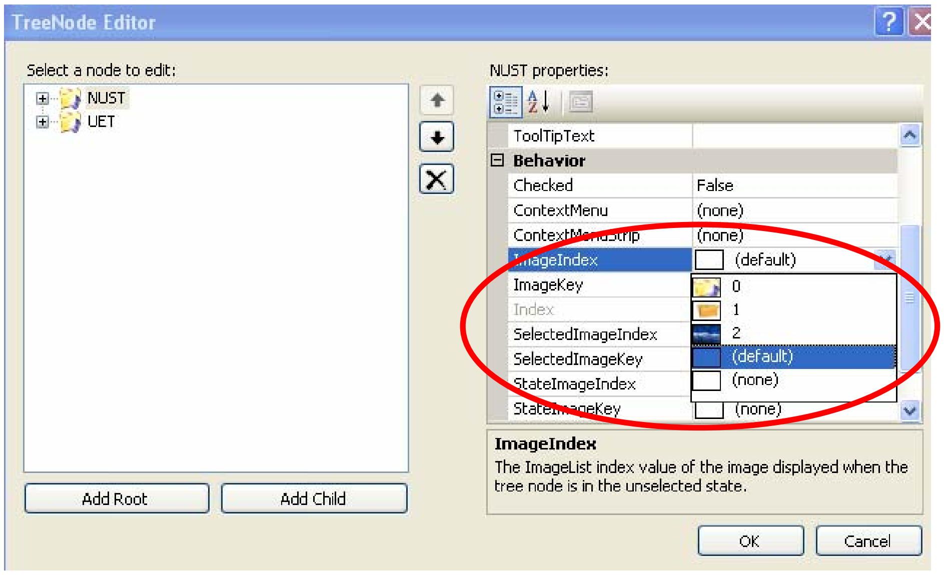
```
NUST
  MCS
    CS
      BESE14
        NC Jehanzaib
        NC Naila
        PC Waqqas
    EE
  SEECS
  EME
UET
```

Node0 properties:

Appearance	
BackColor	
ForeColor	
Name	Node0
NodeFont	(none)
Text	NUST
ToolTipText	
Behavior	
Checked	False
ContextMenu	(none)
ContextMenuStrip	(none)
ImageIndex	(default)

Text
The text displayed in the label of the tree node.

- Use the property ImageIndex of each Node to select an Image that will be associated with the Node.



TreeNode Editor

Select a node to edit:

- NUST
- UET

NUST properties:

ToolTipText	
Behavior	
Checked	False
ContextMenu	(none)
ContextMenuStrip	(none)
ImageIndex	(default)
ImageKey	0
Index	1
SelectedImageIndex	2
SelectedImageKey	(default)
StateImageIndex	(none)
StateImageKey	(none)

ImageIndex
The ImageList index value of the image displayed when the tree node is in the unselected state.

- You can create a Tree and Add a button to the Form. Once clicked, the button displays the selected Node. Following could be used to get the selected item from the Tree

```
TreeView1.SelectedNode.Text
```

Events

- An event is something that happens. In programming terminology an event is when something special happens. These events are so special that they are built in to the programming language. VB.NET has numerous Events that you can write code for. And we're going to explore one of them in this section.

Exploring the Click Event

- Buttons have the ability to be clicked on. When you click a button, the event that is fired is the **Click Event**. If you were to add a button to a form, and then double clicked it, you would see the following code:

```
Private Sub Button1_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) _
Handles Button1.Click
```

```
End Sub
```

- This is a Private Subroutine. The name of the Sub is Button1_Click. The Event itself is at the end: Button1.Click. The Handles word means that this Subroutine can Handle the Click Event of Button1. Without the arguments in the round brackets, the code is this:

```
Private Sub Button1_Click( ) Handles Button1.Click
```

- You can have this Button1_Click Sub Handle other things, too. It can Handle the Click Event of other Buttons, for example. Try this.
  - Add a second button to your Form
  - Double click Button1 to bring up the code
  - At the end of the first line for the Button, add this:

```
Handles Button1.Click, Button2.Click
```

- Add a message box as the code for the Button. Your code window might then look like this:

```
Private Sub Button1_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) _
Handles Button1.Click, Button2.Click

    MessageBox.Show("Handles both the buttons")


 End Sub
```

- Run your programme, and then click both of the buttons in turn. The same message box appears, regardless of which one you clicked.
- The reason it did so was because the Events that the Button1.Click Subroutine can Handle are at the end: the Events for Button1.Click AND Button2.Click.
- You can add as many Events as you want on the End. As long as the Subroutine can Handle them, the Event will happen.

Event Arguments

- The arguments for a Button's click event, the ones from the round brackets, are these two:

```
ByVal sender As System.Object, ByVal e As System.EventArgs
```

- This sets up two variables: one called sender and one called e. Instead of sender being an integer or string variable, the type of variable set up for sender is System.Object. This stores a reference to a control (which button was clicked, for example).
- For the e variable, this is holding an object, too - information about the event. For a button, this information might be which Mouse Button was clicked or where the mouse pointer was on the screen.
- We will explore these arguments and other events (e.g. Keyboard events) in detail in a later lab activity.

Task 2

Employing the controls that you have seen during the last few lab sessions, design a user interface for an **Apartment/house search**. The user may have the option to search an apartment/house for rent or purchase. You may provide options in terms of city, type (apartment or house), budget, number of rooms, furnished/non-furnished or any other options of your own choice. Once the design is complete, just display the selected options in a MessageBox as always.