Drag and Drop

You have probably used the Drag-n-Drop function in many applications, but have not yet added it to your own.

It's not difficult, but there are several components that all have to be coded before it will work. Here's a summary:

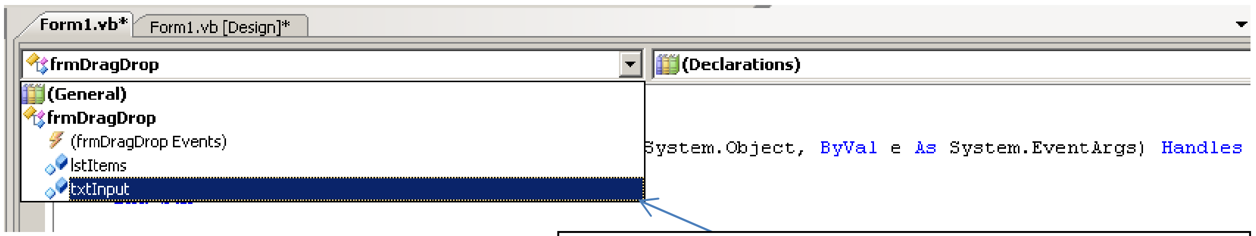
Step 1:

Change the ***AllowDrop*** property of the *target* (the item that will be dropped on) to **True**.
-this property will allow a control (eg. A text box, PictureBox, …) to receive a drop
-it can be changed in the properties menu or in the Form load event

Step 2:

Select the *source* of the drag-drop and add a '**MouseDown**' event. This will copy the value from the source so that it can be dragged elsewhere.

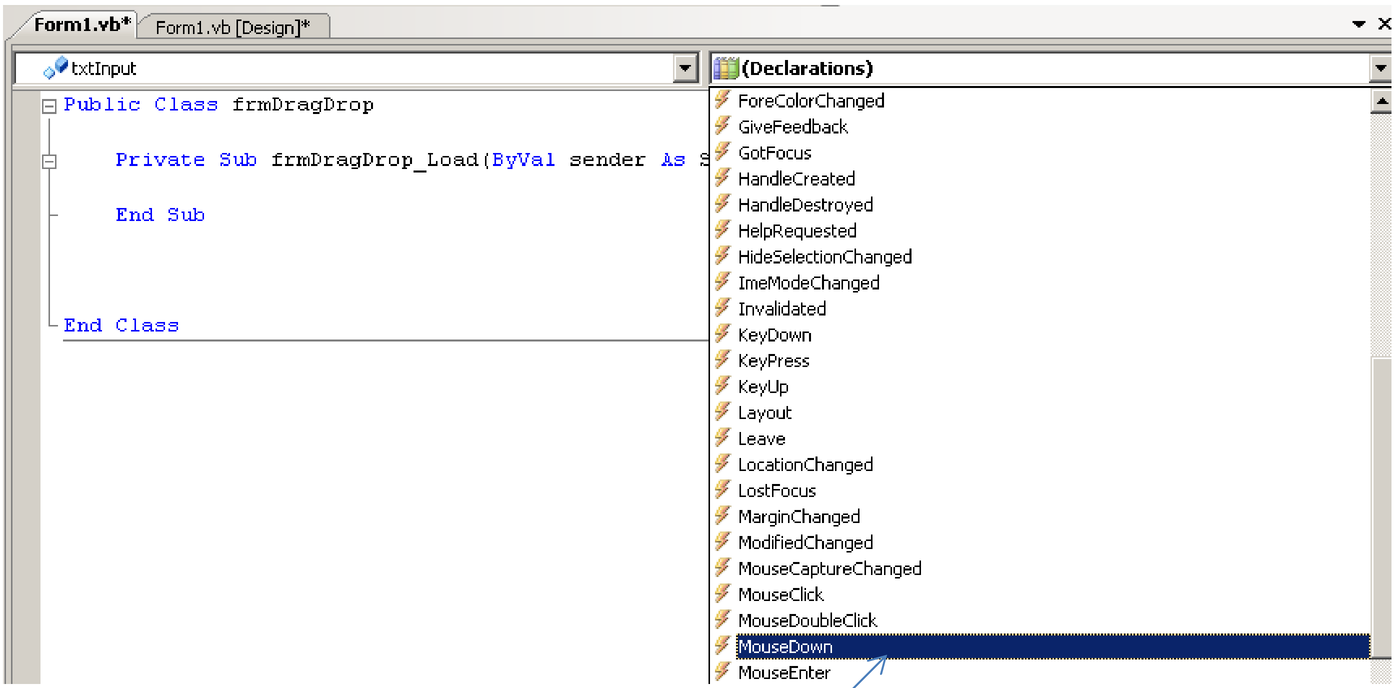
First the **source** object must be selected:



```
System.Object, ByVal e As System.EventArgs) Handles
```

txtInput is where the text that will be dragged comes from (source).

Second the **MouseDown** event is selected so that we can add the appropriate code:

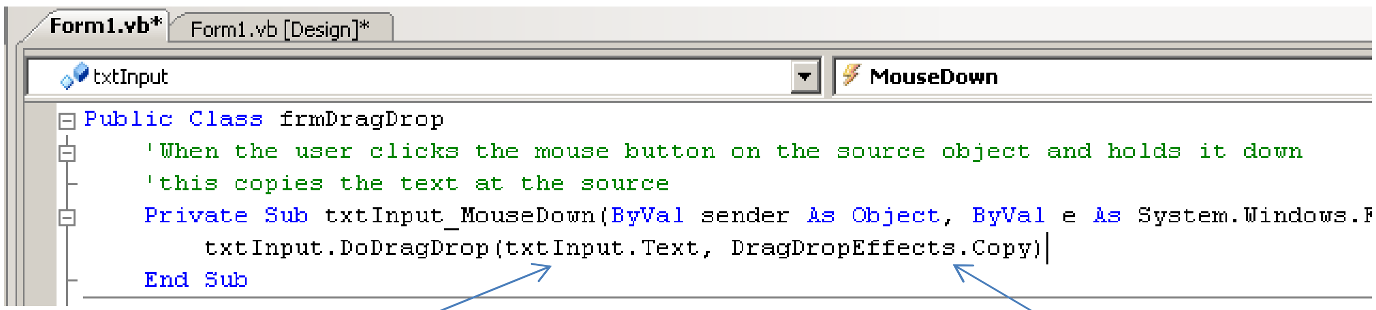


```
Public Class frmDragDrop

    Private Sub frmDragDrop_Load(ByVal sender As S

    End Sub


End Class
```

MouseDown is the event we want to set for the source (when the user clicks and holds the mouse button on the item they wish to drag).

Here is the code that you must enter in the **MouseDown** event of the source:

Add code to the *source* that calls the ***DoDragDrop*** method. This method has **two** parameters. The first is the data that will be dragged and the second sets the value of an enumeration, *DragDropEffects*, that tells the source what kind of drag and drop will be done.



```
Public Class frmDragDrop
    'When the user clicks the mouse button on the source object and holds it down
    'this copies the text at the source
    Private Sub txtInput_MouseDown(ByVal sender As Object, ByVal e As System.Windows.F
        txtInput.DoDragDrop(txtInput.Text, DragDropEffects.Copy)
    End Sub
```

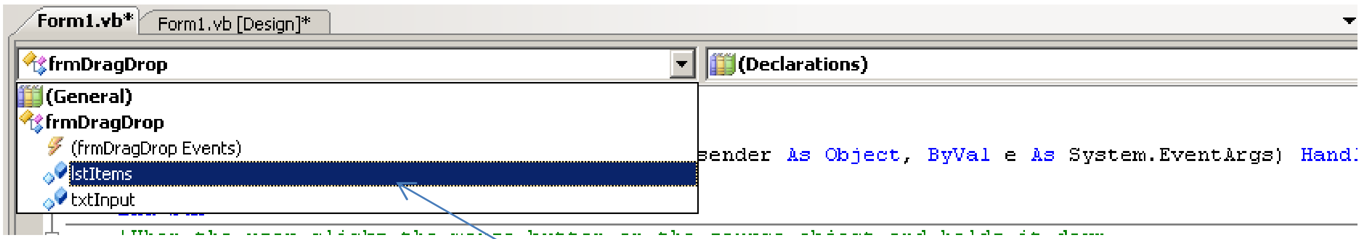
Source of data that will be dragged. The source can be anything, not necessarily a property of the control.

Value of enumeration **e** defines what is happening to the item being dragged (in this case, it is just being copied).

Step 3:

Select the *target* and add the code that will add the item being dragged (text from the source) to the place where it will be 'dropped' (list of items in the target)

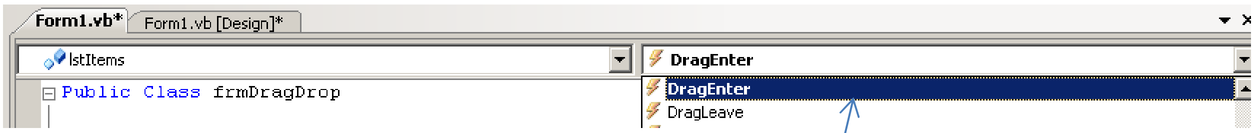
First, the **target** object must be selected (in this case a **List** object is the target)



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

lstItems is the list that we will drop the text onto (target).

Second, the **DragEnter** event is selected so that we can add the appropriate code:



```
Public Class frmDragDrop
```

DragEnter is the event we want to set for the target (when something is dragged over the object).

Add code to the *target*, usually in the *DragEnter* event to set the target Effect property of the **e** parameter.

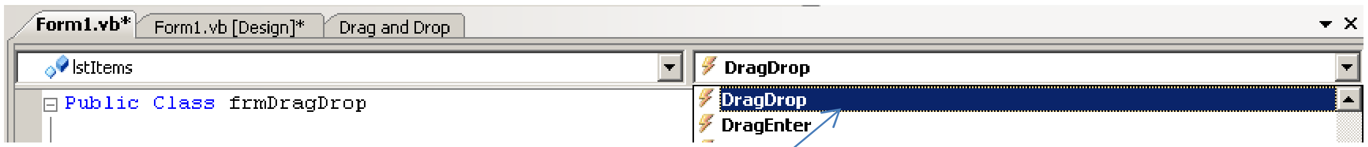
```
'When the item being dragged is dragged over the place you intend to drop it this prepares it
'to be dropped by saving its value into 'e'
Private Sub lstItems_DragEnter(ByVal sender As Object, ByVal e As System.Windows.Forms.DragEventArgs) Ha
    e.Effect = DragDropEffects.Copy
End Sub
```

e refers to the item that has been dragged.

Value of enumeration:
-in this case the data is being copied when it is dragged over the target

Step 4:

Select the target and add the code that will finally 'drop' the object in the destination.



```
Public Class frmDragDrop
```

DragDrop is the event that will define what happened when an object is dropped onto the target.

Add code to the *target* to complete the drag and drop operation in the ***DragDrop*** event of the target.

```
'When the mose button is released (the item is dropped) it takes tha data from 'e' and converts it
'and places it in the list box
Private Sub lstItems_DragDrop(ByVal sender As Object, ByVal e As System.Windows.Forms.DragEventArg
    lstItems.Items.Add(e.Data.GetData(DataFormats.Text).ToString)
End Sub
```

Target object

Items.Add() is how you add items to a List Box.

This retrieves the value from **e** (in this case it is text).

Converts the text to a String variable so it can be added to **lstItems**

Some notes:

The value of the enumeration **DragDropEffects** at both the *source* and the *target* must match for the operation to be allowed. (They're both Copy in step 2 and 3)

You can modify the object before it's dropped, and that can be a valuable technique (if it is a string you could modify it; make it all capitalized for example)

For example, let's make the text capitalized when we drop it:

All we need to do is apply the ToUpper() method to the string we retrieved from **e**.

```
lstItems.Items.Add(e.Data.GetData(DataFormats.Text).ToString().ToUpper())
```

This retrieves the value from **e** (in this case it is text).

Converts the text to a string variable so it can be added to **lstItems**

ToUpper() will return an uppercase version of the text.

Any target that has **AllowDrop** set to True will accept a **DragDrop**. If you have a complicated system where more than one *source* might be available, it's a good idea to include more code to prevent someone trying to drop a picture into a text box.

*This code in the list box **DragEnter** event will do the trick:*

```
If e.Data.GetDataPresent(DataFormats.Text) Then
    e.Effect = DragDropEffects.Copy
Else
    e.Effect = DragDropEffects.None
End If
```

Makes sure that text is going to be dropped and not something else.

What if I want to drag and drop something other than text?

Open up the Drag n Drop Pictures example program. Keep in mind the 3 components of a drag and drop.

1 – The Mouse Down event

Here is where you decide what is to be stored in e when an item is dragged out of a source. (remember that **e** is the object that will store whatever it is we want to be dragged and dropped). We can store whatever we want in **e**. You do not necessarily have to store what was dragged from e. For example, if the source of the drag is a PictureBox, you don't necessarily have to store the image data. You could store text, or something else.

2 – The Drag Enter event

When an object that can be dropped onto has something dragged overtop of itself, you can decide whether the data stored in **e** is of the right type. For example, if you want your item to accept drops of text, you can verify that the data in e is text and not something else. If it does match the type of data you want, you can prepare it for copying.

3 – The Drop event

When something is dropped onto an object, all of its data is stored in **e**. Depending of the type of data that **e** contains, you may be able to examine it and have your program act accordingly depending on its values. For instance, you can have a drop event do anything you want like change the image in a PictureBox, disable a Button, close your program, or anything else you can think of.

Things to Try:

See if you can update the Drag and Drop Pictures program to do the following things:
-Add in the ability to drag the Cat label to make the picture in the PictureBox a cat.
-Have several PictureBoxes with cartoon characters in them. These can be sources. Have a PictureBox with an image of a Microphone in it that will be the destination of the drop. When a character is dropped onto the microphone, pay a sound clip of that character.

Other things to try:

Make your own matching game or a puzzle game.

Notes:

Our tutorial only covers how to drag and drop Text data. You can allow your program to allow a user to drag and drop a variety of data types. The example