



Drag and Drop between list boxes - Beginner's Tutorial



Nish Sivakumar, 13 May 2002



4.72 (107 votes)

An introduction to drag and drop in .NET



Download source files - 5Kb



Download demo - 3Kb

Preamble

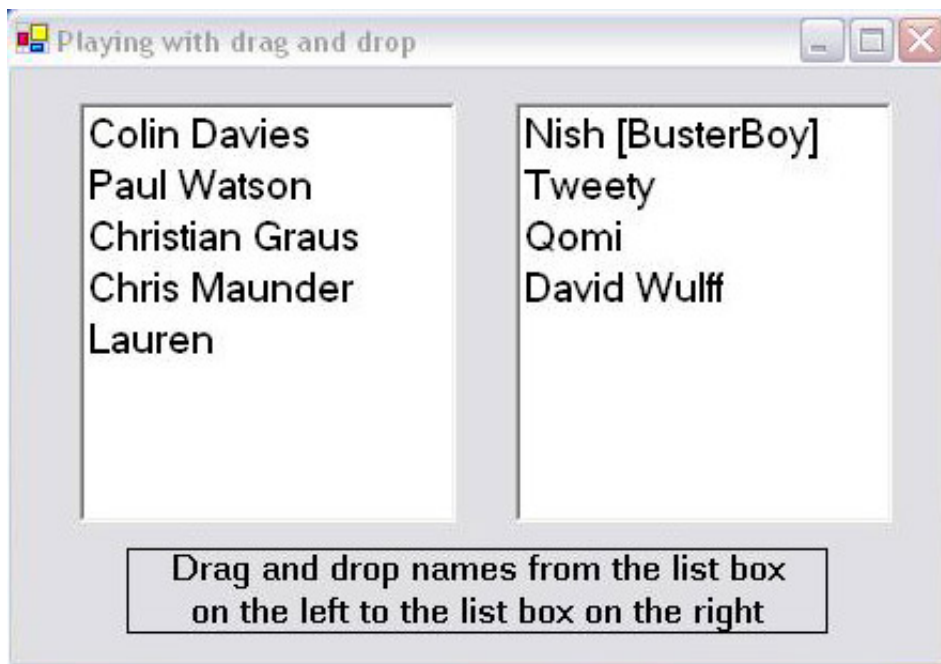
Yeah, you are not alone when you dream of dragging and dropping. I did dream too. It's such an infinite ecstasy when you feel the power of the mouse. Just clicking on an item, dragging it to some other window or control and to simply drop it there gives you a feeling of supremacy. Alright, I guess I better stop with this juvenile gobbledygook and start talking serious stuff. Just today morning I was thinking that it should be quite easy to drag and drop in .NET because that's what I have basically encountered whenever I tried to do something with .NET. Everything is always so much more easier than when you are doing normal SDK/MFC stuff or the Non-dot-Net stuff.

I created a simple C# Windows Forms application. Then I added two list boxes using the form designer. Now my intention was to fill up the list box on the left with some text. By the time I complete the small sample program, I should be able to drag an item from the list box on the left and to drop it on the list box on the right. It should have taken me only about an hour or so, but I was really stuck at one point and luckily for me, James T Johnson offered a word of advice and solved all my problems. He really is an amazing character, that JTJ, you would need to get really close to feel it though.

Screenshot

Well, before we proceed you can ogle at the screen shot made on *ahem* Windows XP Professional. I know

most of you are thinking now that this guy seems to be crazy if he thinks using XP is a big deal. But alas, little do they know the tribulations faced by a 3rd worlder such as myself. In fact it's a bit of a good fortune that I am not still using DOS 5.0. Yeah, things are that bad.



Let's go ahead

The control from which we drag the item is called as a drag-drop source. In our case this would be the list box on the left. And similarly the control where we'll finally drop the dragged item is called the drag-drop target, which for us would be the list box on the right. For those of you who haven't dragged and dropped before [specially people coming from a Linux background], it's quite easy to accomplish. Click on an item, now without raising your finger off the mouse button drag the mouse cursor to some other point, and release the mouse button, whereby you drop the item on that precise point. Make sure you drop it on a drag-drop target control or window, otherwise the whole exercise would be meaningless.

The first thing we need to do is to make sure that the right list box is a drop target. Controls have a property called **AllowDrop** which we need to set to true. You can do this using the form designer or manually via code.

```
this.listBox2.AllowDrop = true;
```

Now we need to add a handler for the **MouseDown** event for the left list box. A **MouseDown** event occurs when you click the mouse anywhere within the list box control.

```
this.listBox1.MouseDown += new MouseEventHandler(  
    this.listBox1_MouseDown);
```

Well, the **MouseDown** handler is simplicity itself. First we figure out which item is directly under the mouse, when it is clicked. For this purpose we use the **IndexFromPoint** function which turned out to be just what I wanted. This was where James was of immense help to me. He pointed out this nice little function to me just when I was groping like a drunken cat in deep black darkness. Well I simply cannot stop singing JTJ's praises today. Pardon me folks.

Once we have the text that is to be dragged, we call the **DoDragDrop** function. This function takes two parameters.

```
public DragDropEffects DoDragDrop(object data,
    DragDropEffects allowedEffects);
```

The first argument is the object that will be dragged out of the drag-drop source. In our case this will be the text of the item in the list box that is directly under the mouse. The second argument is a **DragDropEffects** enumeration. I am showing you the various values possible in the table below. Of course you can also look it up in MSDN. I am using **DragDropEffects.All** cause that's what we are doing. We copy the data to the target and remove it from the source. We need to check the value returned by **DoDragDrop** and if it is equal to **DragDropEffects.All**, this means that the item was safely dropped on a valid target and thus we remove the item from our list box. Otherwise we should not remove it as it means that the dropping was done in a meaningless area.

DragDropEffects Enumeration

Member Name	Description
All	The data is copied, removed from the drag source, and scrolled in the drop target.
Copy	The data is copied to the drop target.
Link	The data from the drag source is linked to the drop target.
Move	The data from the drag source is moved to the drop target.
None	The drop target does not accept the data.
Scroll	Scrolling is about to start or is currently occurring in the drop target.

```
private void listBox1_MouseDown(
    object sender, System.Windows.Forms.MouseEventArgs e)
{
    if(listBox1.Items.Count==0)
        return;

    int index = listBox1.IndexFromPoint(e.X,e.Y);
    string s = listBox1.Items[index].ToString();
    DragDropEffects dde1=DoDragDrop(s,
        DragDropEffects.All);

    if(dde1 == DragDropEffects.All )
    {
        listBox1.Items.RemoveAt(listBox1.IndexFromPoint(e.X,e.Y));
    }
}
```

Alright, now that we have setup the drag-drop source we need to work on the drag-drop target which for us is the listbox on the right. There are four events associated with a drag-drop target. The events are **DragEnter**, **DragOver**, **DragDrop** and **DragLeave**. A **DragEnter** occurs when the mouse pointer has dragged something into the control's client area. A **DragLeave** occurs if it is dragged out of the control's client area without dropping the item. We won't concern ourselves with those two events.

The **DragOver** event occurs after the **DragEnter** event and we need to signal our readiness to accept the

dropped item. We do this by setting the **Effect** property of the **DragEventArgs** object passed to the handler to one of the **DragDropEffects** enumerations. If we set this to **DragDropEvents.None**, then we have essentially rejected the drop.

```
private void listBox2_DragOver(  
    object sender, System.Windows.Forms.DragEventArgs e)  
{  
    e.Effect=DragDropEffects.All;  
}
```

The **DragDrop** event occurs if the mouse is released on top of our control's client area. If we have not signaled our readiness to accept the drop, then instead of a **DragDrop** event a **DragLeave** event occurs. We are passed a **DragEventArgs** object as our second parameter. This has a **IDataObject** member called **Data**. We call the **GetDataPresent** member function on it to verify if the data format in the data is what we are expecting. In our case we are expecting a string. So we check with **DataFormats.StringFormat**. Then once we are sure the data is in the expected format we call **GetData** to retrieve our string and add it to our list box.

```
private void listBox2_DragDrop(  
    object sender, System.Windows.Forms.DragEventArgs e)  
{  
    if(e.Data.GetDataPresent(DataFormats.StringFormat))  
    {  
        string str= (string)e.Data.GetData(  
            DataFormats.StringFormat);  
  
        listBox2.Items.Add(str);  
    }  
}
```

That's about it I guess. This is just a minimal introduction to dragging and dropping. But you can play around with it more. Feel free to post your comments on the article forum, but do not mail me directly, specially not to my busterboy.org email address. It's not a fast server for me and I do not enjoy popping too many mails from there. Thank You.

License

This article, along with any associated source code and files, is licensed under [The Code Project Open License \(CPOL\)](#)

About the Author

Nish Sivakumar



United States 

Nish is a real nice guy who has been writing code since 1990 when he first got his hands on an 8088 with 640 KB RAM. Originally from sunny Trivandrum in India, he has been living in various places over the past few years and often thinks it's time he settled down somewhere.


Nish has been a Microsoft Visual C++ MVP since October, 2002 - awfully nice of Microsoft, he thinks. He maintains an MVP tips and tricks web site - www.voidnish.com where you can find a consolidated list of his articles, writings and ideas on VC++, MFC, .NET and C++/CLI. Oh, and you might want to check out his blog on C++/CLI, MFC, .NET and a lot of other stuff - blog.voidnish.com.

Nish loves reading Science Fiction, P G Wodehouse and Agatha Christie, and also fancies himself to be a decent writer of sorts. He has authored a romantic comedy [Summer Love and Some more Cricket](#) as well as a programming book – [Extending MFC applications with the .NET Framework](#).

Nish's latest book [C++/CLI in Action](#) published by Manning Publications is now available for purchase. You can read more about the book on his blog.

Despite his wife's attempts to get him into cooking, his best effort so far has been a badly done omelette. Some day, he hopes to be a good cook, and to cook a tasty dinner for his wife.

Comments and Discussions

 **50 messages** have been posted for this article Visit <http://www.codeproject.com/Articles/2006/Drag-and-Drop-between-list-boxes-Beginner-s-Tutori> to post and view comments on this article, or click [here](#) to get a print view with messages.

