DoDragDrop and MouseUp

Asked 16 years, 4 months ago Modified 8 years, 10 months ago

Viewed 19k times

20

Is there an easy way to ensure that after a drag-and-drop fails to complete, the MouseUp event isn't eaten up and ignored by the framework?

I have found a blog post describing one mechanism, but it involves a good deal of manual bookkeeping, including status flags, MouseMove events, manual "mouse leave" checking, etc. all of which I would rather not have to implement if it can be avoided.

.net winforms events drag-and-drop

Share

Improve this question

Follow

asked Aug 26, 2008 at 23:23

Chris Ammerman

15.3k ● 8 ● 42 ● 42

argh, I've run into this problem too. Pretty annoying if you ask me! – Isaac Bolinger Jan 23, 2011 at 8:06

1 Answer

Sorted by: Highest score (default)

I was recently wanting to put Drag and Drop functionality in my project and I hadn't come across this issue, but I was intrigued and really wanted to see if I could come up with a better method than the one described in the page you linked to. I hope I clearly understood everything you wanted to do and overall I think I succeeded in solving the problem in a much more elegant and simple fashion.

On a quick side note, for problems like this it would be great if you provide some code so we can see exactly what it is you are trying to do. I say this only because I assumed a few things about your code in my solution...so hopefully it's pretty close.

Here's the code, which I will explain below:

```
this.LabelDrag.QueryContinueDrag += new
System.Windows.Forms.QueryContinueDragEventHandler(thi
this.LabelDrag.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.LabelDrag_
this.LabelDrag.MouseUp += new
System.Windows.Forms.MouseEventHandler(this.LabelDrag_

this.LabelDrop.DragDrop += new
System.Windows.Forms.DragEventHandler(this.LabelDrop_D
this.LabelDrop.DragEnter += new
System.Windows.Forms.DragEventHandler(this.LabelMain_D

public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }

    private void LabelDrop_DragDrop(object sender, Dra
    {
```

```
        LabelDrop.Text = e.Data.GetData(DataFormats.Te
    }


    private void LabelMain_DragEnter(object sender, Dr
    {
        if (e.Data.GetDataPresent(DataFormats.Text))
            e.Effect = DragDropEffects.Copy;
        else
            e.Effect = DragDropEffects.None;

    }

    private void LabelDrag_MouseDown(object sender, Mo
    {
        //EXTREMELY IMPORTANT - MUST CALL LabelDrag's
        //Calling the Form's DoDragDrop WILL NOT allow
fire!
        ((Label)sender).DoDragDrop(TextMain.Text, Drag
    }

    private void LabelDrag_MouseUp(object sender, Mous
    {
        LabelDrop.Text = "LabelDrag_MouseUp";
    }

    private void LabelDrag_QueryContinueDrag(object se
QueryContinueDragEventArgs e)
    {
        //Get rect of LabelDrop
        Rectangle rect = new Rectangle(LabelDrop.Locat
Size(LabelDrop.Width, LabelDrop.Height));

        //If the left mouse button is up and the mouse
LabelDrop
        if (Control.MouseButtons != MouseButtons.Left
!rect.Contains(PointToClient(Control.MousePosition)))
        {
            //Cancel the DragDrop Action
            e.Action = DragAction.Cancel;
            //Manually fire the MouseUp event
            LabelDrag_MouseUp(sender, new MouseEventAr
0, Control.MousePosition.X, Control.MousePosition.Y, 0
        }
```

```
    }

}
```

I have left out most of the designer code, but included the Event Handler link up code so you can be sure what is linked to what. In my example, the drag/drop is occuring between the labels LabelDrag and LabelDrop.

The main piece of my solution is using the QueryContinueDrag event. This event fires when the keyboard or mouse state changes after DoDragDrop has been called on that control. You may already be doing this, but it is very important that you call the DoDragDrop method of the control that is your source and not the method associated with the form. Otherwise QueryContinueDrag will NOT fire!

One thing to note is that QueryContinueDrag will actually fire when you release the mouse *on the drop control* so we need to make sure we allow for that. This is handled by checking that the Mouse position (retrieved with the global Control.MousePosition property) is inside of the LabelDrop control rectangle. You must also be sure to convert MousePosition to a point relative to the Client Window with PointToClient as Control.MousePosition returns a *screen relative* position.

So by checking that the mouse is *not* over the drop control and that the mouse button is now *up* we have effectively captured a MouseUp event for the LabelDrag control! :) Now, you could just do whatever processing

you want to do here, but if you already have code you are using in the MouseUp event handler, this is not efficient. So just call your MouseUp event from here, passing it the necessary parameters and the MouseUp handler won't ever know the difference.

Just a note though, as I call DoDragDrop from *within* the MouseDown event handler in my example, this code should *never* actually get a direct MouseUp event to fire. I just put that code in there to show that it is possible to do it.

Hope that helps!

Share Improve this answer

Follow

edited Feb 19, 2016 at 1:43

Leon Bambrick

26.3k ● 9 ● 52 ● 76

answered Aug 27, 2008 at 14:00

Adam Haile

31.3k ● 60 ● 195 ● 290

1 You're idea is great, thanks! But I think your if statement should look like (button may be in something other than the form, got rid of spurious not): if (Control.MouseButtons != MouseButtons.Left && rect.Contains(button.Parent.PointToClient(Control.MousePosition))) – colithium Sep 5, 2012 at 4:02

`//EXTREMELY IMPORTANT - MUST CALL LabelDrag's DoDragDrop method!!` Thank you so much! Been trying to figure this out for hours. – Dan Bechard Aug 1, 2014 at 16:43

Good example, just wanted to make one small suggestion - change LabelMain_DragEnter to LabelDrop_DragEnter.

Thought for a moment that there were three labels involved.
– Ashley Jul 6, 2015 at 19:34

@Adam Haile If you set the `LabelDrop` in a `TableLayoutPanel` and dock it to fill the `Form`, and set a `MenuStrip` on top of the `Form` then at the very pixels above the `LabelDrop` right between the `TableLayoutPanel` and the `MenuStrip` if you release the button there `MouseUp` event will NOT be triggered, why?
– Simple Jun 22, 2022 at 0:08