Articles » Desktop Development » Tabs & Property Pages » Tabs and Property Pages

Drag and Drop Tab Control

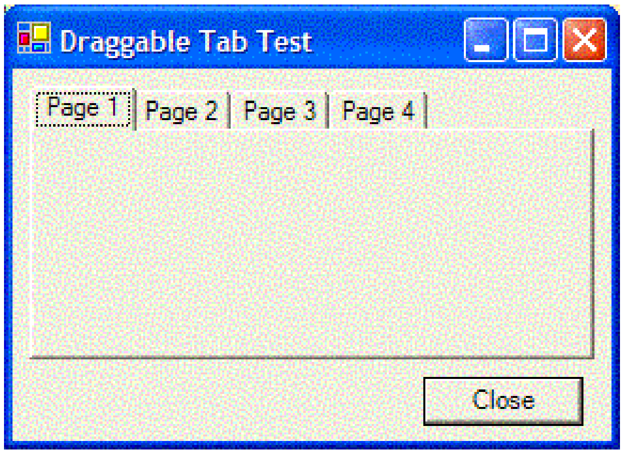
Paul Auger, 15 Jun 2002

★★★★☆ 4.54 (25 votes)

Reorder TabPages in a TabControl through drag and drop.

Download demo project - 14 Kb

Download source - 6 Kb



Introduction

Having a need for my end-users to be able to reorder a set of tabs through drag and drop functionality, I set off on a search to find a control that exhibited this behavior. In fact, I, myself, frequently use this feature within the Visual Studio.NET IDE to better control my documents. Since this ability was not "baked in" to the .NET `TabControl` I took it upon myself to create the control from scratch (or at least extending `TabControl`). The end product of this endeavor is the `DraggableTabControl`

After deriving a class from `TabControl`, the only tricky part, I found, was determining which `TabPage` I was hovering over in the `DragOver` event. When a user drags over the actual tabs of a `TabControl`, the `TabControl` itself actually gets the event, not the `TabPage` To overcome this, I wrote a little function to determine if the point at which the mouse is being dragged is contained within the rectangle of each tab. See the function FindTabPageByTab below.

```
private TabPage GetTabPageByTab(Point pt)
{
    TabPage tp = null;

    //Loop over the tab pages, using GetTabRect() to determine
    //if that pages Tab contains the point we passed in.
    for(int i = 0; i < TabPages.Count; i++)
    {
        if(GetTabRect(i).Contains(pt))
        {
            //We found the tab page, no need to go on...
            tp = TabPages[i];
            break;
        }
    }

    return tp;
}
```

To accomplish the TabPage reordering I first place all of the tabs except the one being dragged into an array and then insert the one I am dragging into the array at the proper point. I then clear all TabPages and add my array back in to the TabControl. It is important, however, that we do this as infrequently as possible to avoid flicker, etc. As a result there are quite a few checks in the code. See the code snippet below to see whay I am talking about...

```
protected override void OnDragOver(System.Windows.Forms.DragEventArgs e)
{
    base.OnDragOver(e);

    Point pt = new Point(e.X, e.Y);
    //We need client coordinates.
    pt = PointToClient(pt);

    //Get the tab we are hovering over.
    TabPage hover_tab = GetTabPageByTab(pt);

    //Make sure we are on a tab.
    if(hover_tab != null)
    {
        //Make sure there is a TabPage being dragged.
        if(e.Data.GetDataPresent(typeof(TabPage)))
        {
            e.Effect = DragDropEffects.Move;
            TabPage drag_tab = (TabPage)e.Data.GetData(typeof(TabPage));

            int item_drag_index = FindIndex(drag_tab);
            int drop_location_index = FindIndex(hover_tab);

            //Don't do anything if we are hovering over ourself.
            if(item_drag_index != drop_location_index)
            {
                ArrayList pages = new ArrayList();

                //Put all tab pages into an array.
                for(int i = 0; i < TabPages.Count; i++)
                {
                    //Except the one we are dragging.
                    if(i != item_drag_index)
                        pages.Add(TabPages[i]);
                }

                //Now put the one we are dragging it at the proper location.
                pages.Insert(drop_location_index, drag_tab);

                //Make them all go away for a nanosec.
                TabPages.Clear();

                //Add them all back in.
                TabPages.AddRange((TabPage[])pages.ToArray(typeof(TabPage)));

                //Make sure the drag tab is selected.
                SelectedTab = drag_tab;
            }
        }
    }
    else
    {
        e.Effect = DragDropEffects.None;
    }
}
```

This class is fairly straight-forward, and should be fully inheritable and delegatable. To use it, you may build it into it's own assembly, or add it to a control library that you may already have. You can add it as a project to an existing solution, or simply add the .cs file to an existing project. Note that if you simply add the .cs file to a project, you also need to add the *DraggableTabControl.bmp* file to the project or get rid of the `[ToolboxBitmap]` attribute. If you make it into its own assembly or compile it with an existing control library, you may add it to the Toolbox by right-clicking it and selecting "Customize Toolbox..." and browsing to it under the ".NET Components" tab.

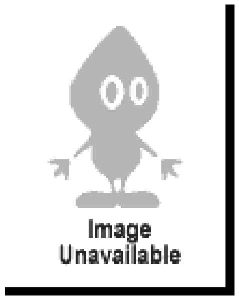
Other than that, the commented code pretty much speaks for itself. I have included the `DraggableTabControl` class source code in full below.

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Windows.Forms;

namespace DraggableTabControl
{
    [ToolboxBitmap(typeof(DraggableTabControl))]
    ///
    /// Summary description for DraggableTabPage.
    ///
    public class DraggableTabControl : System.Windows.Forms.TabControl
    {
        ///
        /// Required designer variable.
        ///
        private System.ComponentModel.Container components = null;

        public DraggableTabControl()
        {
            // This call is required by the Windows.Forms Form Designer.
            InitializeComponent();

            // TODO: Add any initialization after the InitForm call

        }
        ///
        /// Clean up any resources being used.
        ///
        protected override void Dispose( bool disposing )
        {
            if( disposing )
            {
                if(components != null)
                {
                    components.Dispose();
                }
            }
            base.Dispose( disposing );
        }

    #region Component Designer generated code
        ///
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        ///
        private void InitializeComponent()
        {
        }
    #endregion

        protected override void OnDragOver(System.Windows.Forms.DragEventArgs e)
        {
            base.OnDragOver(e);

            Point pt = new Point(e.X, e.Y);
            //We need client coordinates.
            pt = PointToClient(pt);

            //Get the tab we are hovering over.
            TabPage hover_tab = GetTabPageByTab(pt);

            //Make sure we are on a tab.
            if(hover_tab != null)
            {
                //Make sure there is a TabPage being dragged.
                if(e.Data.GetDataPresent(typeof(TabPage)))
                {
                    e.Effect =   DragDropEffects.Move;
                    TabPage drag_tab = (TabPage)e.Data.GetData(typeof(TabPage));
                    int item_drag_index = FindIndex(drag_tab);
                    int drop_location_index= FindIndex(hover_tab);

                    //Don't do anything if we are hovering over ourself.
                    if(item_drag_index ! = drop_location_index)  {

                        ArrayList pages = new ArrayList();
                        //Put all tab pages into an array.
                        for(int i = 0;  i <  TabPages.Count; i++)
                        {
                            //Except the one we are dragging.
                            if(i != item_drag_index)
                                pages.Add(TabPages[i]);
                        }

                        //Now put the one we are dragging it at the proper location.
                        pages.Insert(drop_location_index, drag_tab);

                        //Make them all go away for a nanosec.
                        TabPages.Clear();
                        //Add them all back in.
                        TabPages.AddRange((TabPage[])pages.ToArray(typeof(TabPage)));

                        //Make sure the drag tab is selected.
                        SelectedTab = drag_tab;
                    }
                }
            }
            else
            {
                e.Effect = DragDropEffects.None;
            }
        }

        protected override void OnMouseDown(MouseEventArgs e)
        {
            base.OnMouseDown(e);

            Point pt = new Point(e.X, e.Y);
            TabPage tp = GetTabPageByTab(pt);

            if(tp != null)
            {
                DoDragDrop(tp, DragDropEffects.All);
            }
        }

        ///
        /// Finds the TabPage whose tab is contains the given point.
        ///
        /// The point (given in client coordinates) to look for a TabPage.
        /// The TabPage whose tab is at the given point (null if there isn't one).
        private TabPage GetTabPageByTab(Point pt)
        {
            TabPage tp =

                null;
            for(int i =

                0;  i <  TabPages.Count; i++)
            {
                if(GetTabRect(i).Contains(pt))
                {
                    tp = TabPages[i];
                    break;
                }
            }

            return tp;
        }

        ///
        /// Loops over all the TabPages to find the index of the given TabPage.
        ///
        /// The TabPage we want the index for.
        /// The index of the given TabPage(-1 if it isn't found.)
        private int FindIndex(TabPage page)
        {
            for(int i = 0; i < TabPages.Count; i++)
            {
                if(TabPages[i] == page)
                    return i;
            }
```

```
                return -1;
            }
        }
    }
```

License

About the Author



Paul Auger
Web Developer
United States

No Biography provided

Comments and Discussions

30 messages have been posted for this article Visit **http://www.codeproject.com/Articles/2445/Drag-and-Drop-Tab-Control** to post and view comments on this article, or click **here** to get a print view with messages.