# C# 控件实现内容拖动(DragDrop)功能

<http://www.cnblogs.com/wangshenhe/archive/2012/05/08/2490193.html>

## ****一、将控件内容拖到其他控件****

　　在开发过程中，经常会有客户要求，拖动一个控件的数据到另外一个控件中。例如将其中一个ListBox中的数据拖到另一个ListBox中。或者将DataGridView中的数据拖动到TreeView的某个节点。

　　在应用程序中，是通过处理一系列事件，如DragEnter,DragLeave和DragDrop事件来实现在Windows应用程序中的拖放操作的。通过使用这些事件参数中的可用信息，可以轻松实现拖放操作。

　 　拖放操作在代码中是通过三步实现的，首先是启动拖放操作，在需要拖动数据的控件上实现MouseDown事件响应代码，并调用DoDragDrop() 方法；其次是实现拖放效果，在目标控件上添加DragEnter事件响应代码，使用DragDropEffects枚举类型实现移动或复制等拖动效果；最 后是放置数据操作，在目标控件上添加DragDrop响应代码，把数据添加到目标控件中。

1 using System;

2 using System.Drawing;

3 using System.Collections;

4 using System.ComponentModel;

5 using System.Windows.Forms;

6 using System.Data;

7

8 namespace DragDrop

9 {

10 /// <summary>

11 /// Form1 的摘要说明。

12 /// </summary>

13 public class Form1 : System.Windows.Forms.Form

14 {

15 private System.Windows.Forms.ListBox listBox1;

16 private System.Windows.Forms.ListBox listBox2;

17 /// <summary>

18 /// 必需的设计器变量。

19 /// </summary>

20 private System.ComponentModel.Container components = null;

21

22 public Form1()

23 {

24 //

25 // Windows 窗体设计器支持所必需的

26 //

27 InitializeComponent();

28

29 //

30 // TODO: 在 InitializeComponent 调用后添加任何构造函数代码

31 //

32 }

33

34 /// <summary>

35 /// 清理所有正在使用的资源。

36 /// </summary>

37 protected override void Dispose(bool disposing)

38 {

39 if (disposing)

40 {

41 if (components != null)

42 {

43 components.Dispose();

44 }

45 }

46 base.Dispose(disposing);

47 }

48

49 #region Windows 窗体设计器生成的代码

50 /// <summary>

51 /// 设计器支持所需的方法 - 不要使用代码编辑器修改

52 /// 此方法的内容。

53 /// </summary>

54 private void InitializeComponent()

55 {

56 this.listBox1 = new System.Windows.Forms.ListBox();

57 this.listBox2 = new System.Windows.Forms.ListBox();

58 this.SuspendLayout();

59 //

60 // listBox1

61 //

62 this.listBox1.ItemHeight = 12;

63 this.listBox1.Location = new System.Drawing.Point(32, 24);

64 this.listBox1.Name = "listBox1";

65 this.listBox1.Size = new System.Drawing.Size(120, 280);

66 this.listBox1.TabIndex = 0;

67 this.listBox1.MouseDown += new System.Windows.Forms.MouseEventHandler(this.listBox1\_MouseDown);

68 //

69 // listBox2

70 //

71 this.listBox2.ItemHeight = 12;

72 this.listBox2.Location = new System.Drawing.Point(248, 24);

73 this.listBox2.Name = "listBox2";

74 this.listBox2.Size = new System.Drawing.Size(120, 280);

75 this.listBox2.TabIndex = 0;

76 this.listBox2.DragDrop += new System.Windows.Forms.DragEventHandler(this.listBox2\_DragDrop);

77 this.listBox2.DragEnter += new System.Windows.Forms.DragEventHandler(this.listBox2\_DragEnter);

78 //

79 // Form1

80 //

81 this.AutoScaleBaseSize = new System.Drawing.Size(6, 14);

82 this.ClientSize = new System.Drawing.Size(408, 333);

83 this.Controls.Add(this.listBox1);

84 this.Controls.Add(this.listBox2);

85 this.Name = "Form1";

86 this.Text = "Form1";

87 this.Load += new System.EventHandler(this.Form1\_Load);

88 this.ResumeLayout(false);

89

90 }

91 #endregion

92

93 private void Form1\_Load(object sender, System.EventArgs e)

94 {

95 this.listBox1.AllowDrop = true;

96 this.listBox2.AllowDrop = true;

97 this.listBox1.Items.Add("a");

98 this.listBox1.Items.Add("b");

99 this.listBox1.Items.Add("c");

100 }

101

102 private void listBox1\_MouseDown(object sender, System.Windows.Forms.MouseEventArgs e)

103 {

104 this.listBox1.DoDragDrop(this.listBox1.Items[this.listBox1.SelectedIndex], DragDropEffects.Move);

105 }

106

107 private void listBox2\_DragEnter(object sender, System.Windows.Forms.DragEventArgs e)

108 {

109 if (e.Data.GetDataPresent(DataFormats.Text))

110 {

111 e.Effect = DragDropEffects.Move;

112 }

113 }

114

115 private void listBox2\_DragDrop(object sender, System.Windows.Forms.DragEventArgs e)

116 {

117 this.listBox2.Items.Add(e.Data.GetData(DataFormats.Text));

118 this.listBox1.Items.Remove(e.Data.GetData(DataFormats.Text));

119 }

120 }

121 }

## **二、将文件拖到控件中获得文件路径**

把文件或者目录直接拖放到你的程序上，这种效果用户体验不错。

得到拖过来的路径的代码：（System.Array)e.Data.GetData(DataFormats.FileDrop)。

然后你可以根据这些路径复制粘贴了。

1 using System;

2 using System.Collections.Generic;

3 using System.ComponentModel;

4 using System.Data;

5 using System.Drawing;

6 using System.Linq;

7 using System.Text;

8 using System.Windows.Forms;

9

10 namespace TestFileDrag

11 {

12 public partial class Form1 : Form

13 {

14 public Form1()

15 {

16 InitializeComponent();

17 }

18

19 private void Form1\_Load(object sender, EventArgs e)

20 {

21 SetCtrlDrag.SetCtrlDragEvent(this.textBox1);

22 }

23 }

24

25 public class SetCtrlDrag

26 {

27 public static void SetCtrlDragEvent(Control ctrl)

28 {

29 if(ctrl is TextBox)

30 {

31 TextBox tb = ctrl as TextBox;

32 tb.AllowDrop = true;

33 tb.DragEnter += (sender, e) =>

34 {

35 e.Effect = DragDropEffects.Link;//拖动时的图标

36 };

37 tb.DragDrop += (sender, e) =>

38 {

39 ((TextBox)sender).Text = ((System.Array)e.Data.GetData(DataFormats.FileDrop)).GetValue(0).ToString();

40 };

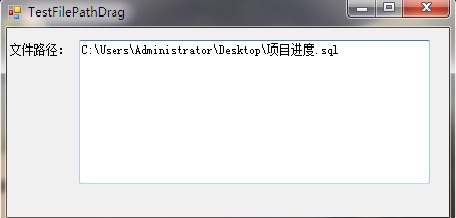
41 }

42 }

43 }

44 }

界面效果



## **三、相关说明**

msdn:<http://msdn.microsoft.com/zh-cn/library/system.windows.forms.dragdropeffects.aspx>

### **1．方法**

实现拖放效果时,C#中提供了一个系统方法DoDragDrop方法，用于实现开始拖放操作，该方法由Control类所定义，由于控件均直接或是间接派生 于Control类，因此开发人员可以在任何可视化组件中调用DoDragDrop方法。DoDragDrop方法使用语法如下：

public DragDropEffects DoDragDrop ( Object data,DragDropEffects allowedEffects)

data：用户所要拖动的数据内容。必须将所要拖动的内容传入到这个方法的第一个参数位置。

allowedEffects：DragDropEffects枚举值之一，此类型包含了拖动操作的效果。DragDropEffects枚举值如表32.8所示。

DragDropEffects枚举值：

枚举值 说明

All 从拖动源复制、移除数据，并将其滚动到放置目标中

Copy 将数据复制到放置目标

Link 将拖动源中的数据链接到放置目标

Move 将拖动源的数据移动到放置目标

None 放置目标不接受该数据

Scroll 即将在放置目标中开始滚动，或当前正在滚动

开发人员在使用DoDragDrop方法时，必须指定参数allowedEffects为表中的任何一个成员，另外，还可以使用位运算符，把其中的任何一个成员作为一个完整参数传入，以得到所需的拖动效果，实现关键代码如下：

DragDropEffects.Copy| DragDropEffects.None

### **2．事件**

C#中提供了一个系统拖放事件，与拖放方法一起使用来达到更好的效果。常用的拖放事件如表所示。

表拖放事件

名称说明

DragEnter 当用户在拖放操作过程中首次将鼠标光标拖到控件上时，会引发该事件

DragDrop 在完成拖放操作时发生

GiveFeedback 在执行拖动操作期间发生

DragLeave 如果用户移出一个窗口，则引发DragLeave事件

DragOver 如果鼠标移动但停留在同一个控件中，则引发DragOver事件

QueryContinueDrag 在拖放操作过程中，当键盘或鼠标按钮状态发生变化时，会引发QueryContinueDrag 事件。QueryContinueDrag事件允许拖动源确定是否应取消拖放操作

## **四、其他博客**

## C#中控件的拖拽操作

<http://blog.sina.com.cn/s/blog_6c6b2acd0100n2hk.html>

下面描述与拖放操作相关的事件的引发方式以及引发时间。

DoDragDrop 方法确定当前光标位置下的控件。然后它将检查该控件是否是有效的放置目标。

如果该控件是有效的放置目标，则 GiveFeedback 事件以指定的拖放效果引发。有关拖放效果的列表，请参见 DragDropEffects 枚举。

跟踪鼠标光标位置、键盘状态和鼠标按钮状态的更改。

如果鼠标移出一个控件，则引发该控件 DragLeave 事件。

如果鼠标进入另一个控件，则引发该控件的 DragEnter 事件。

如果鼠标移动但停留在同一个控件中，则引发 DragOver 事件。

如果更改了键盘或鼠标按钮状态(经过调试发现其实每次鼠标的移动也会引发QueryContinueDrag这个事件)，则引发 QueryContinueDrag 事件并根据事件的QueryContinueDragEventArgs 的 Action 属性值确定是否继续拖动、放置数据或取消操作。

如果 DragAction 的值为 Continue，将引发 DragOver 事件以继续该操作，并引发 GiveFeedback 事件，同时产生新效果，以便能够设置适当的可视反馈。有关有效放置效果的列表，请参见 DragDropEffects 枚举。

注意：

DragOver 和 GiveFeedback 事件成对出现，从而当鼠标从放置目标上移动时，就能够为用户提供有关鼠标位置的最新反馈。

如果 DragAction 的值为 Drop，则放置效果值将返回到源，以便源应用程序可以对源数据执行适当的操作；例如，如果是移动操作，则剪切数据。

如果 DragAction 的值为 Cancel，则引发 DragLeave 事件。

综上所述，我们发现一个简单的拖拽过程需要处理的事件也是非常多的，做软件要细心，这除了敏锐的头脑外还需要一种态度。

开始拖拽 --> (loop (QueryContinueDrag, GiveFeedback)) --> 结束

当然这期间可能需要处理DragLeave、DragEnter、DragOver，一般情况 DragEnter 是必须的，而如果需要随时捕获鼠标在目的控件上的位置以改变目的控件的显示状态时，也需要处理DragOver事件。

DoDragDrop 会直到拖拽结束后才向下执行。这个非常关键!!

示例代码:

using System;

using System.Drawing;

using System.Windows.Forms;

using System.Diagnostics;

namespace WindowsFormsApplication10

{

public class Form1 : System.Windows.Forms.Form

{

private System.Windows.Forms.ListBox ListDragSource;

private System.Windows.Forms.ListBox ListDragTarget;

private System.Windows.Forms.CheckBox UseCustomCursorsCheck;

private System.Windows.Forms.Label DropLocationLabel;

private int indexOfItemUnderMouseToDrag;

private int indexOfItemUnderMouseToDrop;

private Rectangle dragBoxFromMouseDown;

private Point screenOffset;

private Cursor MyNoDropCursor;

private Cursor MyNormalCursor;

/// The main entry point for the application.

[STAThread]

static void Main()

{

Application.Run(new Form1());

}

public Form1()

{

this.ListDragSource = new System.Windows.Forms.ListBox();

this.ListDragTarget = new System.Windows.Forms.ListBox();

this.UseCustomCursorsCheck = new System.Windows.Forms.CheckBox();

this.DropLocationLabel = new System.Windows.Forms.Label();

this.SuspendLayout();

// ListDragSource

this.ListDragSource.Items.AddRange(new object[] {"one", "two", "three", "four",

"five", "six", "seven", "eight",

"nine", "ten"});

this.ListDragSource.Location = new System.Drawing.Point(10, 17);

this.ListDragSource.Size = new System.Drawing.Size(120, 225);

this.ListDragSource.MouseDown += new System.Windows.Forms.MouseEventHandler(this.ListDragSource\_MouseDown);

this.ListDragSource.QueryContinueDrag += new System.Windows.Forms.QueryContinueDragEventHandler(this.ListDragSource\_QueryContinueDrag);

this.ListDragSource.MouseUp += new System.Windows.Forms.MouseEventHandler(this.ListDragSource\_MouseUp);

this.ListDragSource.MouseMove += new System.Windows.Forms.MouseEventHandler(this.ListDragSource\_MouseMove);

this.ListDragSource.GiveFeedback += new System.Windows.Forms.GiveFeedbackEventHandler(this.ListDragSource\_GiveFeedback);

// ListDragTarget

this.ListDragTarget.AllowDrop = true;

this.ListDragTarget.Location = new System.Drawing.Point(154, 17);

this.ListDragTarget.Size = new System.Drawing.Size(120, 225);

this.ListDragTarget.DragOver += new System.Windows.Forms.DragEventHandler(this.ListDragTarget\_DragOver);

this.ListDragTarget.DragDrop += new System.Windows.Forms.DragEventHandler(this.ListDragTarget\_DragDrop);

this.ListDragTarget.DragEnter += new System.Windows.Forms.DragEventHandler(this.ListDragTarget\_DragEnter);

this.ListDragTarget.DragLeave += new System.EventHandler(this.ListDragTarget\_DragLeave);

// UseCustomCursorsCheck

this.UseCustomCursorsCheck.Location = new System.Drawing.Point(10, 243);

this.UseCustomCursorsCheck.Size = new System.Drawing.Size(137, 24);

this.UseCustomCursorsCheck.Text = "Use Custom Cursors";

// DropLocationLabel

this.DropLocationLabel.Location = new System.Drawing.Point(154, 245);

this.DropLocationLabel.Size = new System.Drawing.Size(137, 24);

this.DropLocationLabel.Text = "None";

// Form1

this.ClientSize = new System.Drawing.Size(292, 270);

this.Controls.AddRange(new System.Windows.Forms.Control[] {this.ListDragSource,

this.ListDragTarget, this.UseCustomCursorsCheck,

this.DropLocationLabel});

this.Text = "drag-and-drop Example";

this.ResumeLayout(false);

}

private void ListDragSource\_MouseDown(object sender, System.Windows.Forms.MouseEventArgs e)

{

// Get the index of the item the mouse is below.

indexOfItemUnderMouseToDrag = ListDragSource.IndexFromPoint(e.X, e.Y);

if (indexOfItemUnderMouseToDrag != ListBox.NoMatches)

{

// Remember the point where the mouse down occurred. The DragSize indicates

// the size that the mouse can move before a drag event should be started.

Size dragSize = SystemInformation.DragSize;

// Create a rectangle using the DragSize, with the mouse position being

// at the center of the rectangle.

dragBoxFromMouseDown = new Rectangle(new Point(e.X - (dragSize.Width / 2),

e.Y - (dragSize.Height / 2)), dragSize);

}

else

// Reset the rectangle if the mouse is not over an item in the ListBox.

dragBoxFromMouseDown = Rectangle.Empty;

}

private void ListDragSource\_MouseUp(object sender, System.Windows.Forms.MouseEventArgs e)

{

// Reset the drag rectangle when the mouse button is raised.

dragBoxFromMouseDown = Rectangle.Empty;

}

private void ListDragSource\_MouseMove(object sender, System.Windows.Forms.MouseEventArgs e)

{

//注意这里的严密性, 最好不使用e.Button == MouseButtons.Left, 因为e.Button还可能具有其他的属性

if ((e.Button & MouseButtons.Left) == MouseButtons.Left)

{

// If the mouse moves outside the rectangle, start the drag.

//这里做得太好了，防止意外的拖拽，不是说鼠标动了一下就要拖拽!

if (dragBoxFromMouseDown != Rectangle.Empty &&

!dragBoxFromMouseDown.Contains(e.X, e.Y))

{

// Create custom cursors for the drag-and-drop operation.

try

{

MyNormalCursor = new Cursor("3dwarro.cur");

MyNoDropCursor = new Cursor("3dwno.cur");

}

catch

{

// An error occurred while attempting to load the cursors, so use

// standard cursors.

UseCustomCursorsCheck.Checked = false;

}

finally

{

// The screenOffset is used to account for any desktop bands

// that may be at the top or left side of the screen when

// determining when to cancel the drag drop operation.

screenOffset = SystemInformation.WorkingArea.Location;

// Proceed with the drag-and-drop, passing in the list item.

DragDropEffects dropEffect = ListDragSource.DoDragDrop(ListDragSource.Items[indexOfItemUnderMouseToDrag], DragDropEffects.All | DragDropEffects.Link);

//拖拽结束后继续执行!

// If the drag operation was a move then remove the item.

if (dropEffect == DragDropEffects.Move)

{

ListDragSource.Items.RemoveAt(indexOfItemUnderMouseToDrag);

// Selects the previous item in the list as long as the list has an item.

if (indexOfItemUnderMouseToDrag > 0)

ListDragSource.SelectedIndex = indexOfItemUnderMouseToDrag - 1;

else if (ListDragSource.Items.Count > 0)

// Selects the first item.

ListDragSource.SelectedIndex = 0;

}

// Dispose of the cursors since they are no longer needed.

if (MyNormalCursor != null)

MyNormalCursor.Dispose();

if (MyNoDropCursor != null)

MyNoDropCursor.Dispose();

}

}

}

}

private void ListDragSource\_GiveFeedback(object sender, System.Windows.Forms.GiveFeedbackEventArgs e)

{

// Use custom cursors if the check box is checked.

if (UseCustomCursorsCheck.Checked)

{

// Sets the custom cursor based upon the effect.

e.UseDefaultCursors = false;

if ((e.Effect & DragDropEffects.Move) == DragDropEffects.Move)

Cursor.Current = MyNormalCursor;

else

Cursor.Current = MyNoDropCursor;

}

}

private void ListDragTarget\_DragOver(object sender, System.Windows.Forms.DragEventArgs e)

{

// Determine whether string data exists in the drop data. If not, then

// the drop effect reflects that the drop cannot occur.

if (!e.Data.GetDataPresent(typeof(System.String)))

{

e.Effect = DragDropEffects.None;

DropLocationLabel.Text = "None - no string data.";

return;

}

// Set the effect based upon the KeyState.

if ((e.KeyState & (8 + 32)) == (8 + 32) &&

(e.AllowedEffect & DragDropEffects.Link) == DragDropEffects.Link)

{

// KeyState 8 + 32 = CTL + ALT

// Link drag-and-drop effect.

e.Effect = DragDropEffects.Link;

}

else if ((e.KeyState & 32) == 32 &&

(e.AllowedEffect & DragDropEffects.Link) == DragDropEffects.Link)

{

// ALT KeyState for link.

e.Effect = DragDropEffects.Link;

}

else if ((e.KeyState & 4) == 4 &&

(e.AllowedEffect & DragDropEffects.Move) == DragDropEffects.Move)

{

// SHIFT KeyState for move.

e.Effect = DragDropEffects.Move;

}

else if ((e.KeyState & 8) == 8 &&

(e.AllowedEffect & DragDropEffects.Copy) == DragDropEffects.Copy)

{

// CTL KeyState for copy.

e.Effect = DragDropEffects.Copy;

}

else if ((e.AllowedEffect & DragDropEffects.Move) == DragDropEffects.Move)

{

// By default, the drop action should be move, if allowed.

e.Effect = DragDropEffects.Move;

}

else

e.Effect = DragDropEffects.None;

// Get the index of the item the mouse is below.

// The mouse locations are relative to the screen, so they must be

// converted to client coordinates.

indexOfItemUnderMouseToDrop =

ListDragTarget.IndexFromPoint(ListDragTarget.PointToClient(new Point(e.X, e.Y)));

// Updates the label text.

if (indexOfItemUnderMouseToDrop != ListBox.NoMatches)

{

DropLocationLabel.Text = "Drops before item #" + (indexOfItemUnderMouseToDrop + 1);

}

else

DropLocationLabel.Text = "Drops at the end.";

}

private void ListDragTarget\_DragDrop(object sender, System.Windows.Forms.DragEventArgs e)

{

// Ensure that the list item index is contained in the data.

if (e.Data.GetDataPresent(typeof(System.String)))

{

Object item = (object)e.Data.GetData(typeof(System.String));

// Perform drag-and-drop, depending upon the effect.

if (e.Effect == DragDropEffects.Copy ||

e.Effect == DragDropEffects.Move)

{

// Insert the item.

if (indexOfItemUnderMouseToDrop != ListBox.NoMatches)

ListDragTarget.Items.Insert(indexOfItemUnderMouseToDrop, item);

else

ListDragTarget.Items.Add(item);

}

}

// Reset the label text.

DropLocationLabel.Text = "None";

}

private void ListDragSource\_QueryContinueDrag(object sender, System.Windows.Forms.QueryContinueDragEventArgs e)

{

// Cancel the drag if the mouse moves off the form.

ListBox lb = sender as ListBox;

if (lb != null)

{

Form f = lb.FindForm();

// Cancel the drag if the mouse moves off the form. The screenOffset

// takes into account any desktop bands that may be at the top or left

// side of the screen.

if (((Control.MousePosition.X - screenOffset.X) < f.DesktopBounds.Left) ||

((Control.MousePosition.X - screenOffset.X) > f.DesktopBounds.Right) ||

((Control.MousePosition.Y - screenOffset.Y) < f.DesktopBounds.Top) ||

((Control.MousePosition.Y - screenOffset.Y) > f.DesktopBounds.Bottom))

{

e.Action = DragAction.Cancel;

}

}

}

private void ListDragTarget\_DragEnter(object sender, System.Windows.Forms.DragEventArgs e)

{

// Reset the label text.

DropLocationLabel.Text = "None";

}

private void ListDragTarget\_DragLeave(object sender, System.EventArgs e)

{

// Reset the label text.

DropLocationLabel.Text = "None";

}

}

}