**Часть 1: Обзор элемента управления**

1. **Определение**

MonthCalendar представляет элемент управления Windows, который позволяет выбрать дату с помощью визуального календаря на месяц.

[System.ComponentModel.DefaultBindingProperty("SelectionRange")]

public class MonthCalendar : System.Windows.Forms.Control

Пространство имен: System.Windows.Forms

Сборка: System.Windows.Forms.dll

Наследование: Object => MarshalByRefObject => Component => Control => MonthCalendar

Атрибуты: DefaultBindingPropertyAttribute

1. **Пример**

В следующем примере кода отображается форма, содержащая элемент управления MonthCalendar, который отображает один календарный год. В примере показано задание свойств BackColor, ForeColor, TitleBackColor, TitleForeColor, CalendarDimensions и TrailingForeColor для настройки внешнего вида элемента управления "Календарь". Другие свойства, такие как AnnuallyBoldedDates, BoldedDates и MonthlyBoldedDates, задаются, чтобы настроить даты полужирным шрифтом. В примере также задаются свойства FirstDayOfWeek, MaxDate, MinDate и MaxSelectionCount для изменения формата календаря. События DateSelected и DateChanged также обрабатываются, а их состояние отображается в форме.

Листинг 1 – Month Calendar Example

namespace t1

{

public partial class Form1 : Form

{

private System.Windows.Forms.MonthCalendar monthCalendar1;

private System.Windows.Forms.TextBox textBox1;

public Form1()

{

this.textBox1 = new System.Windows.Forms.TextBox();

this.textBox1.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle;

this.textBox1.Location = new System.Drawing.Point(48, 488);

this.textBox1.Multiline = true;

this.textBox1.ReadOnly = true;

this.textBox1.Size = new System.Drawing.Size(824, 32);

// Create the calendar.

this.monthCalendar1 = new System.Windows.Forms.MonthCalendar();

// Set the calendar location.

this.monthCalendar1.Location = new System.Drawing.Point(47, 16);

// Change the color.

this.monthCalendar1.BackColor = System.Drawing.SystemColors.Info;

this.monthCalendar1.ForeColor = System.Drawing.Color.FromArgb(

((System.Byte)(192)), ((System.Byte)(0)), ((System.Byte)(192)));

this.monthCalendar1.TitleBackColor = System.Drawing.Color.Purple;

this.monthCalendar1.TitleForeColor = System.Drawing.Color.Yellow;

this.monthCalendar1.TrailingForeColor = System.Drawing.Color.FromArgb(

((System.Byte)(192)), ((System.Byte)(192)), ((System.Byte)(0)));

// Add dates to the AnnuallyBoldedDates array.

this.monthCalendar1.AnnuallyBoldedDates =

new System.DateTime[] { new System.DateTime(2002, 4, 20, 0, 0, 0, 0),

new System.DateTime(2002, 4, 28, 0, 0, 0, 0),

new System.DateTime(2002, 5, 5, 0, 0, 0, 0),

new System.DateTime(2002, 7, 4, 0, 0, 0, 0),

new System.DateTime(2002, 12, 15, 0, 0, 0, 0),

new System.DateTime(2002, 12, 18, 0, 0, 0, 0)};

// Add dates to BoldedDates array.

this.monthCalendar1.BoldedDates = new System.DateTime[] { new System.DateTime(2002, 9, 26, 0, 0, 0, 0) };

// Add dates to MonthlyBoldedDates array.

this.monthCalendar1.MonthlyBoldedDates =

new System.DateTime[] {new System.DateTime(2002, 1, 15, 0, 0, 0, 0),

new System.DateTime(2002, 1, 30, 0, 0, 0, 0)};

// Configure the calendar to display 3 rows by 4 columns of months.

this.monthCalendar1.CalendarDimensions = new System.Drawing.Size(4, 3);

// Set week to begin on Monday.

this.monthCalendar1.FirstDayOfWeek = System.Windows.Forms.Day.Monday;

// Set the maximum visible date on the calendar to 12/31/2010.

this.monthCalendar1.MaxDate = new System.DateTime(2010, 12, 31, 0, 0, 0, 0);

// Set the minimum visible date on calendar to 12/31/2010.

this.monthCalendar1.MinDate = new System.DateTime(1999, 1, 1, 0, 0, 0, 0);

// Only allow 21 days to be selected at the same time.

this.monthCalendar1.MaxSelectionCount = 21;

// Set the calendar to move one month at a time when navigating using the arrows.

this.monthCalendar1.ScrollChange = 1;

// Do not show the "Today" banner.

this.monthCalendar1.ShowToday = false;

// Do not circle today's date.

this.monthCalendar1.ShowTodayCircle = false;

// Show the week numbers to the left of each week.

this.monthCalendar1.ShowWeekNumbers = true;

// Add event handlers for the DateSelected and DateChanged events

this.monthCalendar1.DateSelected += new System.Windows.Forms.DateRangeEventHandler(this.monthCalendar1\_DateSelected);

this.monthCalendar1.DateChanged += new System.Windows.Forms.DateRangeEventHandler(this.monthCalendar1\_DateChanged);

// Set up how the form should be displayed and add the controls to the form.

this.ClientSize = new System.Drawing.Size(920, 566);

this.Controls.AddRange(new System.Windows.Forms.Control[] { this.textBox1, this.monthCalendar1 });

this.Text = "Month Calendar Example";

}

private void monthCalendar1\_DateSelected(object sender, System.Windows.Forms.DateRangeEventArgs e)

{

// Show the start and end dates in the text box.

this.textBox1.Text = "Date Selected: Start = " +

e.Start.ToShortDateString() + " : End = " + e.End.ToShortDateString();

}

private void monthCalendar1\_DateChanged(object sender, System.Windows.Forms.DateRangeEventArgs e)

{

// Show the start and end dates in the text box.

this.textBox1.Text = "Date Changed: Start = " +

e.Start.ToShortDateString() + " : End = " + e.End.ToShortDateString();

}

}

}

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 1 – Month Calendar Example

1. **Комментарии**

Элемент MonthCalendar управления позволяет пользователю выбрать дату с помощью визуального отображения. Вы можете ограничить выбор даты и времени, задав свойства MinDate и MaxDate.

Вы можете изменить внешний вид части календаря элемента управления, задав свойства ForeColor, Font, TitleBackColor, TitleForeColor, TrailingForeColor и BackColor.

Примечание: элемент MonthCalendar управления поддерживает только григорианские календари.

Элемент управления MonthCalendar рисуется операционной системой, поэтому событие Paint никогда не вызывается. Если необходимо предоставить настраиваемый вид элемента управления MonthCalendar, необходимо переопределить метод OnPrint, вызвать базовую реализацию OnPrint, а затем выполнить настраиваемую покраску.

Если требуется настраиваемое форматирование даты и выбор только одной даты, можно использовать элемент управления DateTimePicker вместо MonthCalendar. Использование DateTimePicker устраняет большую часть необходимости проверки значений даты и времени.

1. **Конструкторы**

|  |  |
| --- | --- |
| [MonthCalendar()](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.monthcalendar.-ctor?view=windowsdesktop-8.0#system-windows-forms-monthcalendar-ctor) | Инициализирует новый экземпляр класса [MonthCalendar](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.monthcalendar?view=windowsdesktop-8.0). |

1. **Свойства**

|  |  |
| --- | --- |
| [BoldedDates](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.monthcalendar.boldeddates?view=windowsdesktop-8.0#system-windows-forms-monthcalendar-boldeddates) | Возвращает или задает массив объектов [DateTime](https://learn.microsoft.com/ru-ru/dotnet/api/system.datetime?view=windowsdesktop-8.0), который определяет, какие непериодические даты будут выводиться полужирным шрифтом. |
| [FirstDayOfWeek](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.monthcalendar.firstdayofweek?view=windowsdesktop-8.0#system-windows-forms-monthcalendar-firstdayofweek) | Возвращает или задает первый день недели, отображаемый в месячном календаре. |
| [MaxDate](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.monthcalendar.maxdate?view=windowsdesktop-8.0#system-windows-forms-monthcalendar-maxdate) | Возвращает или задает максимальную допустимую дату. |
| [MaxSelectionCount](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.monthcalendar.maxselectioncount?view=windowsdesktop-8.0#system-windows-forms-monthcalendar-maxselectioncount) | Возвращает или задает максимальное количество дней, которые могут быть выбраны в элементе управления "календарь на месяц". |
| [MinDate](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.monthcalendar.mindate?view=windowsdesktop-8.0#system-windows-forms-monthcalendar-mindate) | Возвращает или задает минимальную допустимую дату. |
| [SelectionEnd](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.monthcalendar.selectionend?view=windowsdesktop-8.0#system-windows-forms-monthcalendar-selectionend) | Возвращает или задает конечную дату для выбранного диапазона дат. |
| [SelectionRange](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.monthcalendar.selectionrange?view=windowsdesktop-8.0#system-windows-forms-monthcalendar-selectionrange) | Возвращает или задает выбранный диапазон дат для элемента управления "календарь на месяц". |
| [SelectionStart](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.monthcalendar.selectionstart?view=windowsdesktop-8.0#system-windows-forms-monthcalendar-selectionstart) | Возвращает или задает начальную дату для выбранного диапазона дат. |
| [ShowToday](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.monthcalendar.showtoday?view=windowsdesktop-8.0#system-windows-forms-monthcalendar-showtoday) | Возвращает или задает значение, показывающее, отображается ли дата, представляемая свойством [TodayDate](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.monthcalendar.todaydate?view=windowsdesktop-8.0" \l "system-windows-forms-monthcalendar-todaydate), в нижней части элемента управления. |
| [ShowWeekNumbers](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.monthcalendar.showweeknumbers?view=windowsdesktop-8.0#system-windows-forms-monthcalendar-showweeknumbers) | Возвращает или задает значение, указывающее, будет ли элемент управления "календарь на месяц" отображать номера недель (1—52) слева от каждой строки дней. |
| [Text](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.monthcalendar.text?view=windowsdesktop-8.0#system-windows-forms-monthcalendar-text) | Возвращает или задает текст, который должен отображаться в элементе управления [MonthCalendar](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.monthcalendar?view=windowsdesktop-8.0). |
| [TodayDate](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.monthcalendar.todaydate?view=windowsdesktop-8.0#system-windows-forms-monthcalendar-todaydate) | Возвращает или задает значение, используемое элементом управления [MonthCalendar](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.monthcalendar?view=windowsdesktop-8.0) в качестве сегодняшней даты. |

1. **Методы**

|  |  |
| --- | --- |
| [AddAnnuallyBoldedDate(DateTime)](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.monthcalendar.addannuallyboldeddate?view=windowsdesktop-8.0#system-windows-forms-monthcalendar-addannuallyboldeddate(system-datetime)) | Добавляет день, который каждый год выделяется в месячном календаре полужирным шрифтом. |
| [AddBoldedDate(DateTime)](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.monthcalendar.addboldeddate?view=windowsdesktop-8.0#system-windows-forms-monthcalendar-addboldeddate(system-datetime)) | Добавляет день, который будет отображаться полужирным шрифтом в месячном календаре. |
| [AddMonthlyBoldedDate(DateTime)](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.monthcalendar.addmonthlyboldeddate?view=windowsdesktop-8.0#system-windows-forms-monthcalendar-addmonthlyboldeddate(system-datetime)) | Добавляет день, который каждый месяц отображается в месячном календаре полужирным шрифтом. |
| [DoDragDrop(Object, DragDropEffects)](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.control.dodragdrop?view=windowsdesktop-8.0#system-windows-forms-control-dodragdrop(system-object-system-windows-forms-dragdropeffects)) | Начинает операцию перетаскивания.  (Унаследовано от [Control](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.control?view=windowsdesktop-8.0)) |
| [DoDragDrop(Object, DragDropEffects, Bitmap, Point, Boolean)](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.control.dodragdrop?view=windowsdesktop-8.0#system-windows-forms-control-dodragdrop(system-object-system-windows-forms-dragdropeffects-system-drawing-bitmap-system-drawing-point-system-boolean)) | Начинает операцию перетаскивания.  (Унаследовано от [Control](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.control?view=windowsdesktop-8.0)) |
| [GetDisplayRange(Boolean)](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.monthcalendar.getdisplayrange?view=windowsdesktop-8.0#system-windows-forms-monthcalendar-getdisplayrange(system-boolean)) | Возвращает сведения о датах, которые представляют нижнюю и верхнюю границы диапазона дат, отображаемого элементом управления. |
| [RemoveAllAnnuallyBoldedDates()](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.monthcalendar.removeallannuallyboldeddates?view=windowsdesktop-8.0#system-windows-forms-monthcalendar-removeallannuallyboldeddates) | Снимает выделение со всех дат, ежегодно выделяемых полужирным шрифтом. |
| [RemoveAllBoldedDates()](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.monthcalendar.removeallboldeddates?view=windowsdesktop-8.0#system-windows-forms-monthcalendar-removeallboldeddates) | Снимает выделение со всех непериодических дат, выделяемых полужирным шрифтом. |
| [RemoveAllMonthlyBoldedDates()](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.monthcalendar.removeallmonthlyboldeddates?view=windowsdesktop-8.0#system-windows-forms-monthcalendar-removeallmonthlyboldeddates) | Снимает выделение со всех дат, ежемесячно выделяемых полужирным шрифтом. |
| [RemoveAnnuallyBoldedDate(DateTime)](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.monthcalendar.removeannuallyboldeddate?view=windowsdesktop-8.0#system-windows-forms-monthcalendar-removeannuallyboldeddate(system-datetime)) | Удаляет указанную дату из списка дат, ежегодно выделяемых полужирным шрифтом. |
| [RemoveBoldedDate(DateTime)](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.monthcalendar.removeboldeddate?view=windowsdesktop-8.0#system-windows-forms-monthcalendar-removeboldeddate(system-datetime)) | Удаляет указанную дату из списка непериодических дат, выделяемых полужирным шрифтом. |
| [RemoveMonthlyBoldedDate(DateTime)](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.monthcalendar.removemonthlyboldeddate?view=windowsdesktop-8.0#system-windows-forms-monthcalendar-removemonthlyboldeddate(system-datetime)) | Удаляет указанную дату из списка дат, ежемесячно выделяемых полужирным шрифтом. |

1. **События**

|  |  |
| --- | --- |
| [DateChanged](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.monthcalendar.datechanged?view=windowsdesktop-8.0#system-windows-forms-monthcalendar-datechanged) | Происходит при изменении даты, выбранной в элементе управления [MonthCalendar](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.monthcalendar?view=windowsdesktop-8.0). |
| [DateSelected](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.monthcalendar.dateselected?view=windowsdesktop-8.0#system-windows-forms-monthcalendar-dateselected) | Происходит, когда пользователь явно выбирает дату с помощью мыши. |
| [DragDrop](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.control.dragdrop?view=windowsdesktop-8.0#system-windows-forms-control-dragdrop) | Вызывается при завершении операции перетаскивания.  (Унаследовано от [Control](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.control?view=windowsdesktop-8.0)) |
| [DragEnter](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.control.dragenter?view=windowsdesktop-8.0#system-windows-forms-control-dragenter) | Происходит, когда объект перетаскивается в границы элемента управления.  (Унаследовано от [Control](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.control?view=windowsdesktop-8.0)) |
| [DragLeave](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.control.dragleave?view=windowsdesktop-8.0#system-windows-forms-control-dragleave) | Вызывается, когда объект перетаскивается за пределы элемента управления.  (Унаследовано от [Control](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.control?view=windowsdesktop-8.0)) |
| [DragOver](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.control.dragover?view=windowsdesktop-8.0#system-windows-forms-control-dragover) | Происходит, когда объект перетаскивается через границу элемента управления.  (Унаследовано от [Control](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.control?view=windowsdesktop-8.0)) |
| [MouseDown](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.control.mousedown?view=windowsdesktop-8.0#system-windows-forms-control-mousedown) | Происходит при нажатии кнопки мыши, если указатель мыши находится на элементе управления.  (Унаследовано от [Control](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.control?view=windowsdesktop-8.0)) |
| [MouseUp](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.control.mouseup?view=windowsdesktop-8.0#system-windows-forms-control-mouseup) | Происходит при отпускании кнопки мыши, когда указатель мыши находится на элементе управления.  (Унаследовано от [Control](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.control?view=windowsdesktop-8.0)) |

1. **Явные реализации интерфейса**

|  |  |
| --- | --- |
| [IDropTarget.OnDragDrop(DragEventArgs)](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.control.system-windows-forms-idroptarget-ondragdrop?view=windowsdesktop-8.0#system-windows-forms-control-system-windows-forms-idroptarget-ondragdrop(system-windows-forms-drageventargs)) | Вызывает событие [DragDrop](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.control.dragdrop?view=windowsdesktop-8.0" \l "system-windows-forms-control-dragdrop).  (Унаследовано от [Control](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.control?view=windowsdesktop-8.0)) |
| [IDropTarget.OnDragEnter(DragEventArgs)](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.control.system-windows-forms-idroptarget-ondragenter?view=windowsdesktop-8.0#system-windows-forms-control-system-windows-forms-idroptarget-ondragenter(system-windows-forms-drageventargs)) | Вызывает событие [DragEnter](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.control.dragenter?view=windowsdesktop-8.0" \l "system-windows-forms-control-dragenter).  (Унаследовано от [Control](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.control?view=windowsdesktop-8.0)) |
| [IDropTarget.OnDragLeave(EventArgs)](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.control.system-windows-forms-idroptarget-ondragleave?view=windowsdesktop-8.0#system-windows-forms-control-system-windows-forms-idroptarget-ondragleave(system-eventargs)) | Вызывает событие [DragLeave](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.control.dragleave?view=windowsdesktop-8.0" \l "system-windows-forms-control-dragleave).  (Унаследовано от [Control](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.control?view=windowsdesktop-8.0)) |
| [IDropTarget.OnDragOver(DragEventArgs)](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.control.system-windows-forms-idroptarget-ondragover?view=windowsdesktop-8.0#system-windows-forms-control-system-windows-forms-idroptarget-ondragover(system-windows-forms-drageventargs)) | Вызывает событие [DragOver](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.control.dragover?view=windowsdesktop-8.0" \l "system-windows-forms-control-dragover).  (Унаследовано от [Control](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.control?view=windowsdesktop-8.0)) |

**Часть 2: Drag&Drop**

Drag&Drop – это один из главных способов взаимодействия пользователя с компьютером в графическом интерфейсе. Способ интуитивно ясен, поэтому получил широкое распространение.

Функции Drag&Drop применяются для переноса каких-либо данных с одного элемента в другой. В процессе Drag&Drop участвуют два объекта: объект-источник и объект приемник. В источнике происходит захват элемента нажатием кнопки мыши. Чтобы оставить объект в объекте приемнике, следует отпустить кнопку мыши.

1. **Drag and drop**

Перетаскивание ссылается на передачу данных, в которой используется мышь или другое указывающее устройство, чтобы указать источник данных и его назначение. В обычной операции перетаскивания пользователь выбирает объект для передачи путем перемещения указателя мыши на него и удерживая на ней левую или другую кнопку, назначенную для этой цели. Продолжая удерживать кнопку, пользователь инициирует передачу путем перетаскивания объекта в место назначения, который может быть любым контейнером OLE. Перетаскивание обеспечивает точно те же функции, что и копирование, и вставка буфера обмена OLE, но добавляет визуальные отзывы и устраняет необходимость в меню. На самом деле, если приложение поддерживает копирование и вставку буфера обмена, для поддержки перетаскивания требуется немного дополнений.

Во время операции перетаскивания OLE используются три отдельных фрагмента кода.

| **Перетаскивание исходного кода** | **Реализация и использование** |
| --- | --- |
| [Интерфейс IDropSource](https://learn.microsoft.com/ru-ru/windows/desktop/api/OleIdl/nn-oleidl-idropsource) | Реализуется объектом, содержащим перетаскиваемые данные, называемым источником перетаскивания. |
| [Интерфейс IDropTarget](https://learn.microsoft.com/ru-ru/windows/desktop/api/OleIdl/nn-oleidl-idroptarget) | Реализуется объектом, который предназначен для принятия удаления, называемого целевым объектом удаления. |
| [Функция DoDragDrop](https://learn.microsoft.com/ru-ru/windows/desktop/api/Ole2/nf-ole2-dodragdrop) | Реализован OLE и используется для запуска операции перетаскивания. После выполнения операции он упрощает обмен данными между источником перетаскивания и целевым объектом перетаскивания. |

Интерфейсы IDropSource и IDropTarget можно реализовать в контейнере или в объектном приложении. Роль источника перетаскивания или целевого объекта перетаскивания не ограничивается одним типом приложения OLE.

Функция OLE DoDragDrop реализует цикл, который отслеживает перемещение мыши и клавиатуры до тех пор, пока не будет отменено перетаскивание. DoDragDrop — это ключевая функция в процессе перетаскивания, упрощающая взаимодействие между источником перетаскивания и целевым объектом перетаскивания.

Во время операции перетаскивания для пользователя можно отобразить три типа обратной связи.

| **Тип обратной связи** | **Description** |
| --- | --- |
| Отзывы о источнике | Предоставленный источником перетаскивания, исходный отзыв указывает, что данные перетаскиваются и не изменяются во время перетаскивания. Как правило, данные выделены для сигнала о том, что он был выбран. |
| Обратная связь указателя | Предоставленный источником перетаскивания, обратная связь указателя указывает, что происходит, если мышь выпущена в любой момент. Обратная связь указателя постоянно изменяется по мере перемещения мыши и /или нажатия клавиши модификатора. Например, если указатель перемещается в окно, которое не может принять удаление, указатель изменяется на символ "недопустимый". |
| Целевая обратная связь | При условии целевого объекта удаления целевая обратная связь указывает, где происходит удаление. |

1. **Drag source responsibilities**

Источник перетаскивания отвечает за следующие задачи:

* Предоставление объекта передачи данных для целевого объекта удаления, предоставляющего интерфейсы IDataObject и IDropSource.
* Создание указателя и обратной связи источника.
* Определение того, когда операция перетаскивания отменена или произошла операция удаления.
* Выполнение любого действия с исходными данными, вызванными операцией удаления, например удаление данных или создание ссылки на нее.

Основная задача — создание объекта передачи данных, предоставляющего интерфейсы IDataObject и IDropSource. Источник перетаскивания может или не включать копию выбранных данных. В том числе это не обязательно, но это помогает защититься от непреднамеренных изменений и позволяет коду операций буфера обмена быть идентичным коду перетаскивания.

Пока выполняется операция перетаскивания, источник перетаскивания отвечает за настройку указателя мыши и при необходимости для предоставления дополнительных отзывов о источнике пользователю. Источник перетаскивания не может предоставлять обратную связь, которая отслеживает положение мыши, отличное от фактического задания реального указателя (см. функцию SetCursor). Это правило должно быть применено, чтобы избежать конфликтов с отзывом, предоставленным целевым объектом удаления. (Источник перетаскивания также может быть целевым объектом перетаскивания. При удалении на себя исходный или целевой объект может, конечно, предоставить целевую обратную связь для отслеживания положения мыши. Однако в этом случае это целевой объект для отслеживания мыши, а не источник.) На основе отзывов, предлагаемых целевым объектом удаления, источник задает соответствующий указатель.

1. **Пример**

В следующем примере кода демонстрируется операция перетаскивания между несколькими элементами управления ListBox. В примере вызывается метод DoDragDrop при запуске действия перетаскивания. Действие перетаскивания запускается, если мышь переместилась больше, чем SystemInformation.DragSize из расположения мыши во время MouseDown события. Метод IndexFromPoint используется для определения индекса перетаскиваемого элемента во время события MouseDown.

В примере также показано использование пользовательских курсоров для операции перетаскивания. В этом примере требуется, чтобы в каталоге приложения существовали два файла курсоров, 3dwarro.cur и 3dwno.cur, для настраиваемых курсоров перетаскивания и без перетаскивания соответственно. Пользовательские курсоры будут использоваться, если установлен флажок UseCustomCursorsCheckCheckBox. Пользовательские курсоры задаются в обработчике событий GiveFeedback.

Состояние клавиатуры вычисляется в обработчике DragOver событий для правых ListBox, чтобы определить, какая операция перетаскивания будет зависеть от состояния клавиш SHIFT, CTRL, ALT или CTRL+ALT. Расположение в объекте ListBox, где будет происходить удаление, также определяется во время события DragOver. Если данные для удаления не являются String, то для DragEventArgs.Effect задается значение None в DragDropEffects. Наконец, состояние удаления отображается в DropLocationLabelLabel.

Данные для удаления ListBox определяются в обработчике DragDrop событий, а String значение добавляется в соответствующее место в ListBox. Если операция перетаскивания перемещается за пределы формы, то операция перетаскивания отменяется в обработчике QueryContinueDrag событий.

Изображение выглядит как текст, снимок экрана, дисплей, число

Автоматически созданное описание

Рисунок 2 – Drag&Drop SWOT

Листинг 2 – Drag&Drop SWOT

using System;

using System.Drawing;

using System.Reflection.Emit;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

using Button = System.Windows.Forms.Button;

using Label = System.Windows.Forms.Label;

using TextBox = System.Windows.Forms.TextBox;

namespace t2

{

public partial class Form1 : Form

{

private ListBox ListDragSource;

private ListBox ListDragTarget1;

private ListBox ListDragTarget2;

private ListBox ListDragTarget3;

private ListBox ListDragTarget4;

private CheckBox UseCustomCursorsCheck;

private Label DropLocationLabel;

private Label SourceLabel;

private Label Target1Label;

private Label Target2Label;

private Label Target3Label;

private Label Target4Label;

private TextBox NewSource;

private Button AddSourceButton;

private int indexOfItemUnderMouseToDrag;

private int indexOfItemUnderMouseToDrop;

private Rectangle dragBoxFromMouseDown;

private Point screenOffset;

private Cursor MyNoDropCursor;

private Cursor MyNormalCursor;

public Form1()

{

this.ListDragSource = new ListBox();

this.ListDragTarget1 = new ListBox();

this.ListDragTarget2 = new ListBox();

this.ListDragTarget3 = new ListBox();

this.ListDragTarget4 = new ListBox();

this.UseCustomCursorsCheck = new CheckBox();

this.DropLocationLabel = new Label();

this.SourceLabel = new Label();

this.Target1Label = new Label();

this.Target2Label = new Label();

this.Target3Label = new Label();

this.Target4Label = new Label();

this.NewSource = new TextBox();

this.AddSourceButton = new Button();

this.SuspendLayout();

// ListDragSource

this.ListDragSource.Location = new Point(12, 37);

this.ListDragSource.Size = new Size(180, 129);

this.ListDragSource.MouseDown += this.ListDragSource\_MouseDown;

this.ListDragSource.QueryContinueDrag += this.ListDragSource\_QueryContinueDrag;

this.ListDragSource.MouseUp += this.ListDragSource\_MouseUp;

this.ListDragSource.MouseMove += this.ListDragSource\_MouseMove;

this.ListDragSource.GiveFeedback += this.ListDragSource\_GiveFeedback;

// ListDragTarget1

this.ListDragTarget1.AllowDrop = true;

this.ListDragTarget1.Location = new Point(198, 37);

this.ListDragTarget1.Size = new Size(180, 129);

this.ListDragTarget1.DragOver += this.ListDragTarget1\_DragOver;

this.ListDragTarget1.DragDrop += this.ListDragTarget1\_DragDrop;

this.ListDragTarget1.DragEnter += this.ListDragTarget\_DragEnter;

this.ListDragTarget1.DragLeave += this.ListDragTarget\_DragLeave;

// ListDragTarget2

this.ListDragTarget2.AllowDrop = true;

this.ListDragTarget2.Location = new Point(384, 37);

this.ListDragTarget2.Size = new Size(180, 129);

this.ListDragTarget2.DragOver += this.ListDragTarget2\_DragOver;

this.ListDragTarget2.DragDrop += this.ListDragTarget2\_DragDrop;

this.ListDragTarget2.DragEnter += this.ListDragTarget\_DragEnter;

this.ListDragTarget2.DragLeave += this.ListDragTarget\_DragLeave;

// ListDragTarget3

this.ListDragTarget3.AllowDrop = true;

this.ListDragTarget3.Location = new Point(198, 197);

this.ListDragTarget3.Size = new Size(180, 129);

this.ListDragTarget3.DragOver += this.ListDragTarget3\_DragOver;

this.ListDragTarget3.DragDrop += this.ListDragTarget3\_DragDrop;

this.ListDragTarget3.DragEnter += this.ListDragTarget\_DragEnter;

this.ListDragTarget3.DragLeave += this.ListDragTarget\_DragLeave;

// ListDragTarget4

this.ListDragTarget4.AllowDrop = true;

this.ListDragTarget4.Location = new Point(384, 197);

this.ListDragTarget4.Size = new Size(180, 129);

this.ListDragTarget4.DragOver += this.ListDragTarget4\_DragOver;

this.ListDragTarget4.DragDrop += this.ListDragTarget4\_DragDrop;

this.ListDragTarget4.DragEnter += this.ListDragTarget\_DragEnter;

this.ListDragTarget4.DragLeave += this.ListDragTarget\_DragLeave;

// UseCustomCursorsCheck

this.UseCustomCursorsCheck.Location = new Point(10, 358);

this.UseCustomCursorsCheck.Size = new Size(199, 24);

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

// DropLocationLabel

this.DropLocationLabel.Location = new Point(212, 358);

this.DropLocationLabel.Size = new Size(199, 24);

this.DropLocationLabel.Text = "None";

// SourceLabel

SourceLabel.Location = new Point(12, 9);

SourceLabel.Size = new Size(63, 25);

SourceLabel.Text = "Факты";

// Target1Label

Target1Label.Location = new Point(198, 9);

Target1Label.Size = new Size(159, 25);

Target1Label.Text = "Сильные стороны";

// Target2Label

Target2Label.Location = new Point(384, 9);

Target2Label.Size = new Size(149, 25);

Target2Label.Text = "Слабые стороны";

// Target3Label

Target3Label.Location = new Point(198, 169);

Target3Label.Size = new Size(124, 25);

Target3Label.Text = "Возможности";

// Target4Label

Target4Label.Location = new Point(384, 169);

Target4Label.Size = new Size(72, 25);

Target4Label.Text = "Угрозы";

// NewSource

NewSource.Location = new Point(12, 197);

NewSource.Size = new Size(180, 31);

// AddSource

AddSourceButton.Location = new Point(12, 234);

AddSourceButton.Size = new Size(180, 40);

AddSourceButton.Text = "Добавить факт";

AddSourceButton.UseVisualStyleBackColor = true;

AddSourceButton.Click += AddSourceButton\_Click;

// Form1

this.ClientSize = new Size(598, 394);

this.Controls.AddRange(new Control[] {

this.ListDragSource,

this.ListDragTarget1,

this.ListDragTarget2,

this.ListDragTarget3,

this.ListDragTarget4,

this.SourceLabel,

this.Target1Label,

this.Target2Label,

this.Target3Label,

this.Target4Label,

//this.UseCustomCursorsCheck,

//this.DropLocationLabel,

this.NewSource,

this.AddSourceButton});

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

this.ResumeLayout(false);

}

private void AddSourceButton\_Click(object sender, EventArgs e)

{

if (NewSource.Text != "")

{

ListDragSource.Items.Add(NewSource.Text);

NewSource.Text = string.Empty;

}

}

private void ListDragSource\_MouseDown(object sender, 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, MouseEventArgs e)

{

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

dragBoxFromMouseDown = Rectangle.Empty;

}

private void ListDragSource\_MouseMove(object sender, MouseEventArgs e)

{

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

{

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

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);

// 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, 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 ListDragTarget1\_DragOver(object sender, 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 = CTRL + 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)

{

// CTRL 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 =

ListDragTarget1.IndexFromPoint(ListDragTarget1.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 ListDragTarget1\_DragDrop(object sender, DragEventArgs e)

{

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

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

{

Object item = 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)

ListDragTarget1.Items.Insert(indexOfItemUnderMouseToDrop, item);

else

ListDragTarget1.Items.Add(item);

}

}

// Reset the label text.

DropLocationLabel.Text = "None";

}

private void ListDragTarget2\_DragOver(object sender, 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 = CTRL + 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)

{

// CTRL 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 =

ListDragTarget2.IndexFromPoint(ListDragTarget2.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 ListDragTarget2\_DragDrop(object sender, DragEventArgs e)

{

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

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

{

Object item = 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)

ListDragTarget2.Items.Insert(indexOfItemUnderMouseToDrop, item);

else

ListDragTarget2.Items.Add(item);

}

}

// Reset the label text.

DropLocationLabel.Text = "None";

}

private void ListDragTarget3\_DragOver(object sender, 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 = CTRL + 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)

{

// CTRL 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 =

ListDragTarget3.IndexFromPoint(ListDragTarget3.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 ListDragTarget3\_DragDrop(object sender, DragEventArgs e)

{

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

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

{

Object item = 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)

ListDragTarget3.Items.Insert(indexOfItemUnderMouseToDrop, item);

else

ListDragTarget3.Items.Add(item);

}

}

// Reset the label text.

DropLocationLabel.Text = "None";

}

private void ListDragTarget4\_DragOver(object sender, 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 = CTRL + 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)

{

// CTRL 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 =

ListDragTarget4.IndexFromPoint(ListDragTarget4.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 ListDragTarget4\_DragDrop(object sender, DragEventArgs e)

{

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

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

{

Object item = 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)

ListDragTarget4.Items.Insert(indexOfItemUnderMouseToDrop, item);

else

ListDragTarget4.Items.Add(item);

}

}

// Reset the label text.

DropLocationLabel.Text = "None";

}

private void ListDragSource\_QueryContinueDrag(object sender, 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, DragEventArgs e)

{

// Reset the label text.

DropLocationLabel.Text = "None";

}

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

{

// Reset the label text.

DropLocationLabel.Text = "None";

}

}

}

1. **Комментарии**

Параметр allowedEffects определяет, какие операции перетаскивания могут выполняться. Если операция перетаскивания должна взаимодействовать с приложениями в другом процессе, данные должны быть либо базовым управляемым классом (String, Bitmap или Metafile), либо объектом, реализующим ISerializable или IDataObject.

Далее описывается, как и когда возникают события, связанные с операциями перетаскивания.

Метод DoDragDrop определяет элемент управления в текущем расположении курсора. Затем он проверяет, является ли элемент управления допустимым целевым объектом удаления.

Если элемент управления является допустимым целевым объектом перетаскивания, GiveFeedback событие вызывается с указанным эффектом перетаскивания. Список эффектов перетаскивания см. в перечислении DragDropEffects.

Отслеживаются изменения позиции указателя мыши, состояния клавиатуры и кнопки мыши.

* Если пользователь перемещает указатель мыши за пределы окна, происходит событие DragLeave.
* Если указатель мыши перемещается на другой элемент управления, для этого элемента вызывается событие DragEnter.
* При перемещении мыши в пределах одного элемента управления возникает событие DragOver.

При изменении состояния клавиатуры или кнопки мыши возникает событие, QueryContinueDrag которое определяет, следует ли продолжить перетаскивание, удалить данные или отменить операцию на основе значения Action свойства события QueryContinueDragEventArgs.

* Если значение DragAction равно Continue, DragOver событие вызывается для продолжения операции, и GiveFeedback событие вызывается с новым эффектом, чтобы можно было задать соответствующую визуальную обратную связь. Список допустимых эффектов сброса см. в перечислении DragDropEffects.

Примечание: События DragOver и GiveFeedback связаны таким образом, что при перемещении мыши по целевому объекту перетаскивания пользователю предоставляется наиболее актуальная обратная связь о положении мыши.

* Если значение DragAction равно Drop, значение эффекта удаления возвращается источнику, чтобы исходное приложение смогла выполнить соответствующую операцию с исходными данными, например вырезать данные, если операция была перемещением.
* Если значение DragAction равно Cancel, возникает событие DragLeave.

Примечание: Метод DoDragDrop перехватывает все исключения и повторно создает только следующие исключения безопасности или критические исключения: SecurityException; NullReferenceException; StackOverflowException; OutOfMemoryException; ThreadAbortException; ExecutionEngineException; IndexOutOfRangeException; AccessViolationException.