## События мыши и фокуса

В WPF для мыши определены следующие события:

|  |  |  |
| --- | --- | --- |
| **Событие** | **Тип события** | **Описание** |
| **GotMouseCapture** | Поднимающееся | Возникает при получении фокуса с помощью мыши |
| **LostMouseCapture** | Поднимающееся | Возникает при потере фокуса с помощью мыши |
| **MouseEnter** | Прямое | Возникает при вхождении указателя мыши в пределы элемента |
| **MouseLeave** | Прямое | Возникает, когда указатель мыши выходит за пределы элемента |
| **MouseLeftButtonDown** | Поднимающееся | Возникает при нажатии левой кнопки мыши |
| **PreviewMouseLeftButtonDown** | Туннельное | Возникает при нажатии левой кнопки мыши |
| **MouseLeftButtonUp** | Поднимающееся | Возникает при освобождении левой кнопки мыши |
| **PreviewMouseLeftButtonUp** | Туннельное | Возникает при освобождении левой кнопки мыши |
| **MouseRightButtonDown** | Поднимающееся | Возникает при нажатии правой кнопки мыши |
| **PreviewMouseRightButtonDown** | Туннельное | Возникает при нажатии правой кнопки мыши |
| **MouseRightButtonUp** | Поднимающееся | Возникает при освобождении правой кнопки мыши |
| **PreviewMouseRightButtonUp** | Туннельное | Возникает при освобождении правой кнопки мыши |
| **MouseDown** | Поднимающееся | Возникает при нажатии кнопки мыши |
| **PreviewMouseDown** | Туннельное | Возникает при нажатии кнопки мыши |
| **MouseUp** | Поднимающееся | Возникает при освобождении кнопки мыши |
| **PreviewMouseUp** | Туннельное | Возникает при освобождении кнопки мыши |
| **MouseMove** | Поднимающееся | Возникает при передвижении указателя мыши |
| **PreviewMouseMove** | Туннельное | Возникает при передвижении указателя мыши |
| **MouseWheel** | Поднимающееся | Возникает при передвижении колесика мыши |
| **PreviewMouseWheel** | Туннельное | Возникает при передвижении колесика мыши |

Если вдруг мы не хотим, чтобы элемент генерировал события мыши, то мы можем у него установить свойство IsHitTestVisible="False"

Большинство обработчиков событий мыши в качестве параметра получают объект MouseEventArgs, имеющий ряд интересных свойств и методов, которые мы можем использовать:

* **ButtonState**: возвращает состояние кнопки мыши. Хранит одно из значений перечисления **MouseButtonState**:
  + Pressed: кнопка наата
  + Released: кнопка отжата
* **ChangedButton**: получает кнопку, которая ассоциирована с данным событием. Хранит одно из значений перечисления **MouseButton**:
  + Left: левая кнопка мыши
  + Middle: средняя кнопка мыши
  + Right: правая кнопка мыши
  + XButton1: дополнительная кнопка мыши
  + XButton2: дополнительная кнопка мыши
* **ClickCount**: хранит число сделанных нажатий
* **LeftButton**: хранит состояние левой кнопки мыши в виде MouseButtonState
* **MiddleButton**: хранит состояние средней кнопки мыши в виде MouseButtonState
* **RightButton**: хранит состояние правой кнопки мыши в виде MouseButtonState
* **XButton1**: хранит состояние первой дополнительной кнопки
* **XButton2**: хранит состояние второй дополнительной кнопки
* **GetPosition()**: метод, который возвращает координаты нажатия в виде объекта Point

Например, используем метод GetPosition(). Для этого установим для грида обработчик:

|  |  |
| --- | --- |
| 1 | <Grid MouseDown="Grid\_MouseDown"> |

И определим этот обработчик:

|  |  |
| --- | --- |
| 1  2  3  4  5 | private void Grid\_MouseDown(object sender, MouseButtonEventArgs e)  {      Point p = e.GetPosition(this);      MessageBox.Show("Координата x=" +p.X.ToString()+ " y="+p.Y.ToString());  } |

### События перетаскивания

События перетаскивания (drag & drop) связаны с перетаскиванием элементов, когда пользователь, нажимая на элементе мышкой и удерживая мышь нажатой, перемещает указатель на другой элемент, тем самым перемещая на этот элемент ранее нажатый.

|  |  |  |
| --- | --- | --- |
| **Событие** | **Тип события** | **Описание** |
| **DragEnter** | Поднимающееся | Возникает при перетаскивании при вхождении указателя мыши в пределы элемента |
| **DragOver** | Поднимающееся | Возникает при перемещении курсора в пределах границ элемента управления |
| **DragLeave** | Поднимающееся | Возникает при перемещении курсора мыши за пределы элемента |
| **Drop** | Поднимающееся | Возникает при завершении перетаскивания |
| **PreviewDragEnter** | Тунельное | Возникает при перетаскивании при вхождении указателя мыши в пределы элемента |
| **PreviewDragOver** | Тунельное | Возникает при перемещении курсора в пределах границ элемента управления |
| **PreviewDragLeave** | Тунельное | Возникает при перемещении курсора мыши за пределы элемента |
| **PreviewDrop** | Тунельное | Возникает при завершении перетаскивания |

Эти события используют объект **DragEventArgs**, который имеет ряд свойств и методов:

* **GetPosition**: возвращает позицию мыши
* **Data**: объект, представляющий буфер обмена - то есть те данные, которые перемещаются
* **Effects** и **AllowedEffects**: представляют эффект перетаскивния. Хранят одно из значений перечисления **DragDropEffects**:
  + All: данные копируются из источника в целевой элемент с удалением из источника
  + Copy: данные просто копируются из источника в целевой элемент
  + Link: данные из источника связываются с данными из целевого элемента
  + Move: данные перемещаются из источника в целевой элемент
  + None: отсутствие эффекта
  + Scroll: данные прокручиваются при копировании в целевой элемент
* **KeyStates**: хранит значение из перечисления **DragDropKeyStates**, которое указывает, какая клавиша клавиатуры или мыши зажата во время перетаскивания: LeftMouseButton, RightMouseButton, MiddleMouseButton, ShiftKey, ControlKey, AltKey, None

Посмотрим на примере. Допустим, у нас следующая разметка xaml:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | <Window x:Class="EventsApp.MainWindow"          xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"          xmlns:d="http://schemas.microsoft.com/expression/blend/2008"          xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"          xmlns:local="clr-namespace:EventsApp"          mc:Ignorable="d"          Title="MainWindow" Height="250" Width="400">      <Grid>          <Grid.RowDefinitions>              <RowDefinition />              <RowDefinition />          </Grid.RowDefinitions>          <DockPanel >              <TextBox x:Name="textBox1" MouseDown="textBox1\_MouseDown" />          </DockPanel>            <Button x:Name="button1" Grid.Row="1" AllowDrop="True" Drop="button1\_Drop"  />      </Grid>  </Window> |

Здесь мы будем перемещать введенный текст из текстового поля на кнопку. Чтобы кнопка могла принимать перемещаемые объекты, установим ее свойство AllowDrop="True". Одни элементы на другие, то нам надо у элементов-приемников всегда устанавливать данное свойство.

Здесь также подключены два обработчика события, которые мы зададим в коде C#:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | private void textBox1\_MouseDown(object sender, MouseButtonEventArgs e)  {      DragDrop.DoDragDrop(textBox1, textBox1.Text, DragDropEffects.Copy);  }    private void button1\_Drop(object sender, DragEventArgs e)  {      button1.Content = e.Data.GetData(DataFormats.Text);  } |

Чтобы захватить элемент для переноса, нам надо вызвать метод DragDrop.DoDragDrop, который в качестве первого параметра принимает элемент-источник, с которого идет перетаскивание, второй параметр - что перетаскиваем (в данном случае текст), и третий параметр - тип эффекта. Так как в данном случае у нас копирование, то устанавливаем DragDropEffects.Copy. Также мы можем использовать и другие константы: Move, None, Link, Scroll, All.

Введем текст в текстовое поле, выделим его, нажмем левой кнопкой и, не отпуская, переместим курсор в пределы кнопки. И отпустим. Здесь уже возникнет событие Drop кнопки, обработчик которого также прост: мы присваиваем ее содержимому данные перетаскивания. И поскольку мы перетаскиваем текст, то в качестве параметра выставляем **DataFormats.Text**

### **События получения/потери фокуса**

При обработке событий фокуса следует помнить, что элемент может получать фокус только в том случае, если его свойство Focusable имеет значение true.

Чтобы программным способом передать элементу фокус, надо вызвать у него методы Focus или MoveFocus:

|  |  |
| --- | --- |
| 1 | textBox1.Focus(); |

|  |  |  |
| --- | --- | --- |
| **Событие** | **Тип события** | **Описание** |
| **GotFocus** | Поднимающееся | Возникает при получении фокуса |
| **LostFocus** | Поднимающееся | Возникает при потере фокуса |
| **GotKeyboardFocus** | Поднимающееся | Возникает при получении фокуса с помощью клавиатуры |
| **PreviewGotKeyboardFocus** | Туннельное | Возникает при получении фокуса с помощью клавиатуры |
| **LostKeyboardFocus** | Поднимающееся | Возникает при потере фокуса с помощью клавиатуры |
| **PreviewLostKeyboardFocus** | Туннельное | Возникает при потере фокуса с помощью клавиатуры |

Обработем событие получения фокуса для текстового поля:

|  |  |
| --- | --- |
| 1 | <TextBox GotFocus="TextBox\_GotFocus"  /> |

В файле кода пропишем обработчик:

|  |  |
| --- | --- |
| 1  2  3  4 | private void TextBox\_GotFocus(object sender, RoutedEventArgs e)  {      MessageBox.Show("Получение фокуса");  } |