События клавиатуры

Последнее обновление: 22.01.2016

К событиям клавиатуры можно отнести следующие события:

Событие	Тип события	Описание
KeyDown	Поднимающееся	Возникает при нажатии клавиши
PreviewKeyDown	Туннельное	Возникает при нажатии клавиши
KeyUp	Поднимающееся	Возникает при освобождении клавиши
PreviewKeyUp	Туннельное	Возникает при освобождении клавиши
TextInput	Поднимающееся	Возникает при получении элементом текстового ввода (генерируется не только клавиатурой, но и стилусом)
PreviewTextInput	Туннельное	Возникает при получении элементом текстового ввода

Большинство событий клавиатуры (KeyUp/PreviewKeyUp, KeyDown/PreviewKeyDown) принимает в качестве аргумента объект

KeyEventArgs, у которого можно отметить следующие свойства:

- **Key** позволяет получить нажатую или отпущенную клавишу
- **SystemKey** позволяет узнать, нажата ли системная клавиша, например, Alt
- **KeyboardDevice** получает объект KeyboardDevice, представляющее устройство клавиатуры
- **IsRepeat** указывает, что клавиша удерживается в нажатом положении
- **IsUp** и **IsDown** указывает, была ли клавиша нажата или отпущена
- **IsToggled** указывает, была ли клавиша включена - относится только к включаемым клавишам Caps Lock, Scroll Lock, Num Lock

Например, обработаем событие KeyDown для текстового поля и выведем данные о нажатой клавише в текстовый блок:

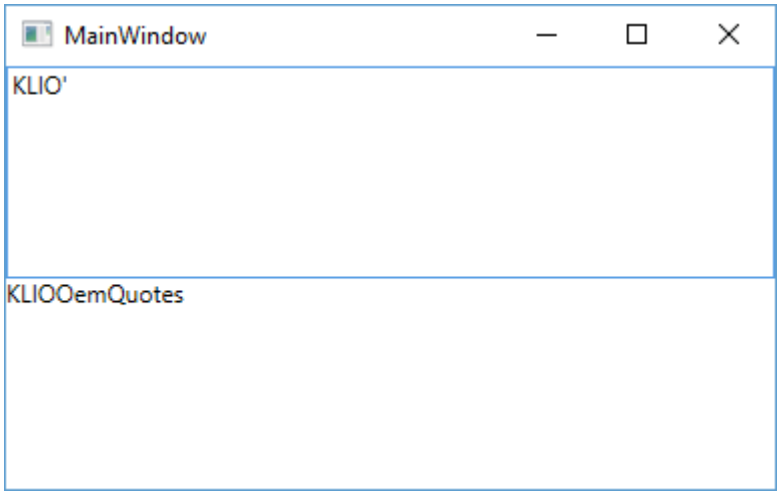
```
<Window x:Class="EventsApp.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:EventsApp"
        mc:Ignorable="d"
        Title="MainWindow" Height="250" Width="400">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition />
            <RowDefinition />
        </Grid.RowDefinitions>
        <DockPanel >
            <TextBox KeyDown="TextBox_KeyDown"  />
        </DockPanel>
        <TextBlock x:Name="textBlock1" Grid.Row="1" />
    </Grid>
</Window>
```

А в файле кода пропишем обработчик `TextBox_KeyDown`:

```
private void TextBox_KeyDown(object sender, KeyEventArgs e)
```

```
{
    textBlock1.Text += e.Key.ToString();
}
```

Здесь в текстовый блок добавляется текстовое представление нажатой клавиши в текстовом поле:



Правда, в данном случае реальную пользу от текстового представления мы можем получить только для алфавитно-цифровых клавиш, в то время как при нажатии специальных клавиш или кавычек будут добавляться их полные текстовые представления, например, для кавычек - OemQuotes.

Если нам надо отловить нажатие какой-то опредленной клавиши, то мы можем ее проверить через перечисление **Key**:

```
if (e.Key == Key.OemQuotes)
    textBlock1.Text += "'"; // добавляем кавычки
else
    textBlock1.Text += e.Key.ToString();
```

Объект `KeyboardDevice` позволяет нам получить ряд дополнительных данных о собыиях клавиатуры через ряд свойств и методов:

- **Modifiers** позволяет узнать, какая клавиша была нажата вместе с основной (Ctrl, Shift, Alt)
- **IsKeyDown()** определяет, была ли нажата определенная клавиша во время события
- **IsKeyUp()** позволяет узнать, была ли отжата определенная клавиша во время события

- **IsKeyToggled()** позволяет узнать, была ли во время события включена клавиша Caps Lock, Scroll Lock или Num Lock
- **GetKeyStates()** возвращает одно из значений перечисления KeyStates, которое указывает на состояние клавиши

Пример использования KeyEventArgs при одновременном нажатии двух клавиш Shift и F1:

```
1  private void TextBox_KeyDown(object sender, KeyEventArgs e)
2  {
3      if (e.KeyboardDevice.Modifiers == ModifierKeys.Shift && e.Key == Key.F1)
4          MessageBox.Show("HELLO");
5  }
```

События TextInput/PreviewTextInput в качестве параметра принимают объект **TextCompositionEventArgs**. Из его свойств стоит отметить, пожалуй, только свойство `Text`, которое получает введенный текст, именно текст, а не текстовое представление клавиши. Для этого добавим к текстовому полю обработчик:

```
1  <TextBox Height="40" Width="260" PreviewTextInput="TextBox_TextInput" />
```

И определим обработчик в файле кода:

```
1  private void TextBox_TextInput(object sender, TextCompositionEventArgs e)
2  {
3      textBlock1.Text += e.Text;
4  }
```

Причем в данном случае я обрабатываю именно событие PreviewTextInput, а не TextInput, так как элемент TextBox подавляет событие TextInput, и вместо него генерирует событие TextChanged. Для большинства других элементов управления, например, кнопок, событие TextInput прекрасно срабатывает.

Валидация текстового ввода

События открывают нам большой простор для валидации текстового ввода. Нередко при вводе используются те или иные ограничения: нельзя вводить цифровые символы или, наоборот, можно только цифровые и т.д. Посмотрим, как мы можем провети валидацию ввода. К примеру, возьмем ввод номера телефона. Сначала зададим обработку двух событий в xaml:

```
1  <TextBox PreviewTextInput="TextBox_PreviewTextInput" PreviewKeyDown="TextBox_PreviewKeyDown"  />
```

И определим в файле кода обработчики:

```
private void TextBox_PreviewTextInput(object sender, TextCompositionEventArgs e)
{
    int val;
    if (!Int32.TryParse(e.Text, out val) && e.Text!="-")
    {
        e.Handled = true; // отклоняем ввод
    }
}

private void TextBox_PreviewKeyDown(object sender, KeyEventArgs e)
{
    if (e.Key == Key.Space)
    {
        e.Handled = true; // если пробел, отклоняем ввод
    }
}
```

Для валидации ввода нам надо использовать обработчики для двух событий - PreviewKeyDown и PreviewTextInput. Дело в том, что нажатия не всех клавиш PreviewTextInput обрабатывает. Например, нажатие на клавишу пробела не обрабтывается. Поэтому также применяется обработка и PreviewKeyDown.

Сами обработчики проверяют ввод и если ввод соответствует критериям, то он отклоняется с помощью установки **e.Handled = true**. Тем самым мы говорим, что событие обработано, а введенные текстовые сиволы не будут появляться в текстовом поле. Конкретно в данном случае пользователь может вводить только цифровые символы и пробел в соответствии с форматом телефонного номера.

Назад **Содержание** **Вперед**

Николай Щеглов • 3 года назад

В последнем предложении ошибка:
"Конкретно в данном случае пользователь может вводить только цифровые символы и пробел в соответствии с форматом телефонного номера."
Там ведь можно вводить только цифровые символы и дефис.

1 • Ответить • Поделиться ›

strelok 372 • 2 года назад • edited

Добрый день, ошибка в примере с двумя нажатиями кнопки:
Вместо

```
private void TextBox_KeyDown(object sender, KeyEventArgs e)
{
if (e.KeyboardDevice.Modifiers == ModifierKeys.Shift && e.Key == Key.F1)
MessageBox.Show("HELLO");
}
```

Надо

```
private void TextBox_KeyDown(object sender, KeyEventArgs e)
{
if (e.KeyboardDevice.Modifiers.HasFlag(ModifierKeys.Shift) && e.Key == Key.F1)
MessageBox.Show("HELLO");
}
```