

XAML

Extensible Application Markup Language

- 1) WPF не нуждается в XAML
- 2) XAML не нуждается в WPF

Код на XAML

```
<Button x:Name="submitButton" FontFamily="Consolas" FontSize="20" Foreground="Green">
    Submit
</Button>
```



Та же кнопка в C#

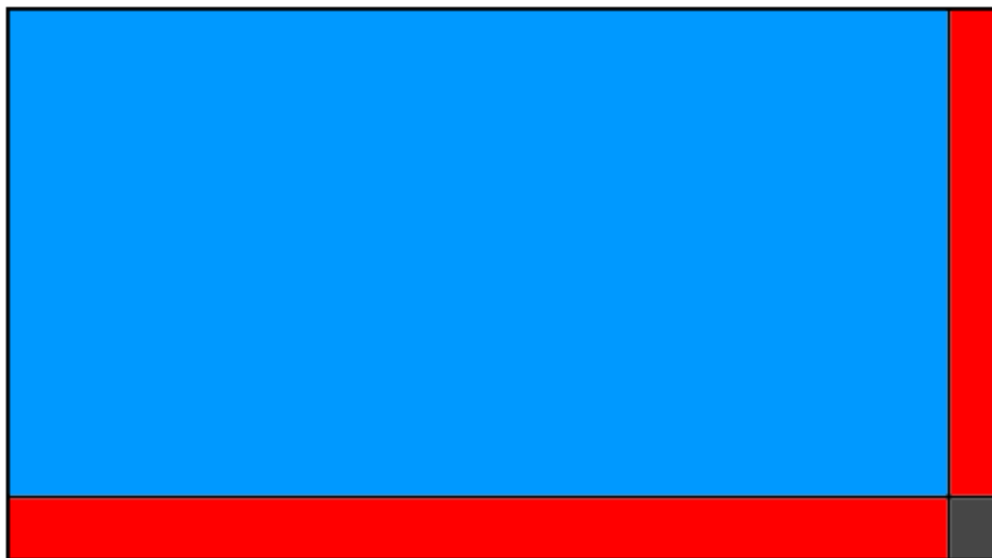
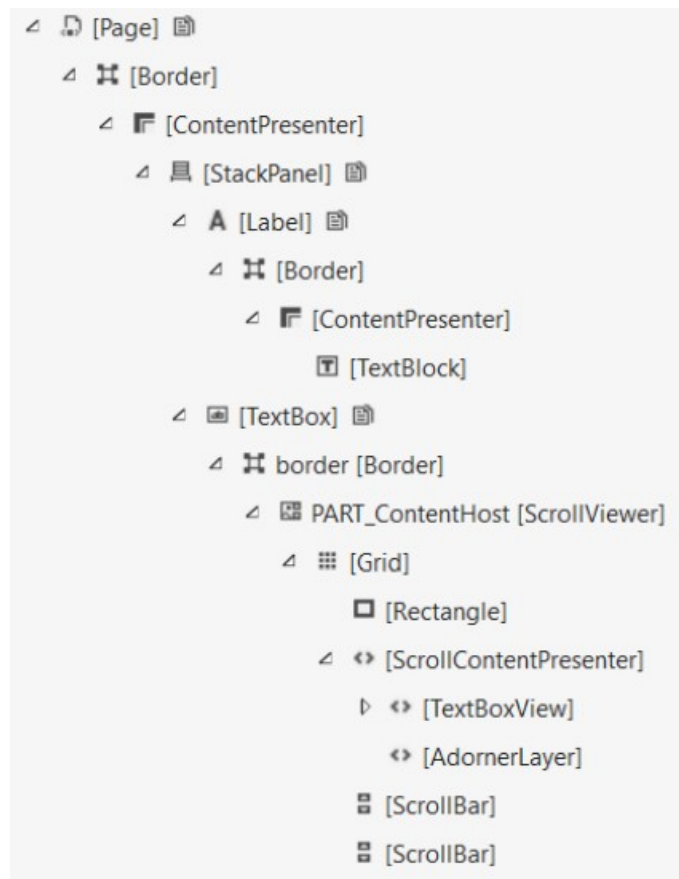
```
var submitButton = new Button
{
    Content = "Submit",
    FontFamily = new FontFamily("Consolas"),
    FontSize = 20.0,
    Foreground = Brushes.Green
};
```

Всё, что может XAML можно сделать используя программный код.

Деревья пользовательского интерфейса

```
<Page>
    <StackPanel Orientation="Horizontal">
        <Label>Message:</Label>
        <TextBox Width="100"/>
    </StackPanel>
</Page>
```





События и команды

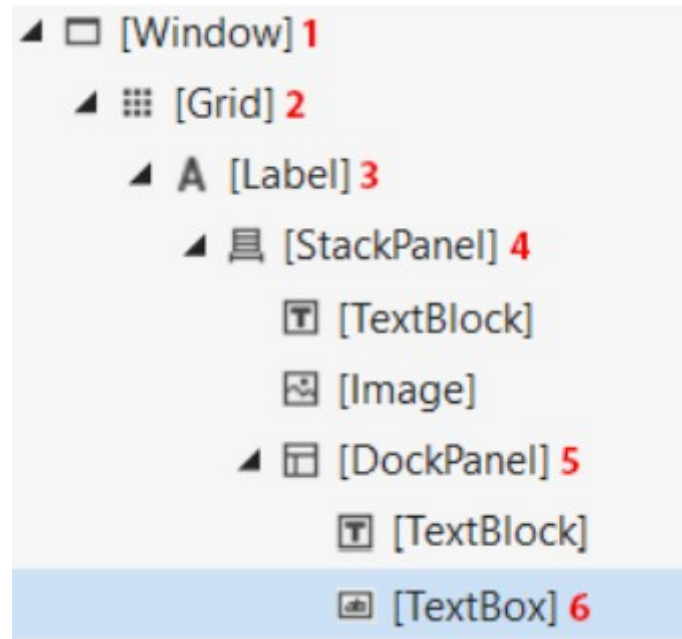
События связаны с визуальным деревом. Они возникают в конкретных элементах пользовательского интерфейса, при взаимодействии с устройствами ввода. Например, клик мыши по элементу или нажатие клавиши клавиатуры в объекте клавиатурного фокуса.

Для того, чтобы не прикреплять обработку событий к каждому конкретному элементу визуального дерева, существует механизм маршрутизации событий.

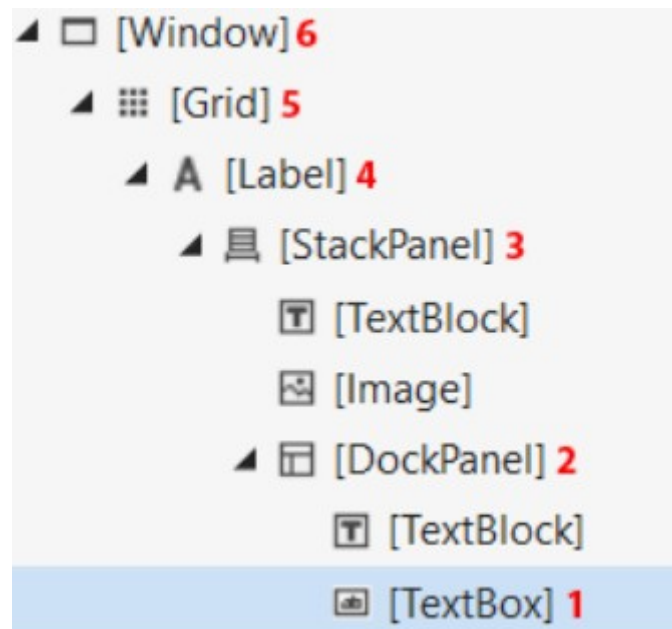
Он позволяет обрабатывать события предками элемента для которого, оно возникло.

Два вида маршрутизации:

туннелирование (tunneling) – событие начинается с корневого элемента и направляется по дочерним, пока не дойдёт до того, которому оно предназначено.



всплытие (bubbling) – наоборот, начинается с элемента, для которого предназначено и всплывает вверх по дереву, пока не найдёт готовый его обработать элемент.



События обычно существуют парно. Есть предварительное событие (preview), оно туннелируется, и есть основное событие, оно всплывает.

События связаны с низкоуровневым взаимодействием — клик мыши, нажатие клавиши на клавиатуре, и т.п.

Но часто требуется более высокий уровень абстракции. Например, нажатие сочетания клавиш Ctrl+P, нажатие кнопки «Печать» и выбор пункта меню «Файл» → «Печать» должен в

любом случае вызывать окно печати на принтер. WPF предоставляет механизм команд, который даёт доступ к такому уровню абстракции.

Элементы управления

Lookless

Элементы управления в WPF не знают, как выглядят. Они не содержат этой информации.

Их вид задаётся с помощью другой сущности — шаблона.

Примитивные элементы

Большинство элементов, которые могут быть отображены на экране, не являются элементами управления.

Существует класс `System.Windows.Controls.Control` – от него наследуются все элементы управления. Элементы управления требуют наличия интерактивного поведения: кнопки, списки, поля для ввода текста, полосы прокрутки и т.д.

Классы же описывающие такие вещи, как базовый прямоугольник, базовый эллипс, изображения и т.п. являются примитивными и наследуются от более базового `System.Windows.FrameworkElement`. Более того, `Control` тоже наследуется от `FrameworkElement`.

Только наследники `Control` имеют шаблон, описывающий их визуализацию.

Наследники `FrameworkElement` не имеют шаблона, а обладают собственной визуализацией.

Панели

Панель — это элемент, который может быть добавлен в визуальное дерево и содержать несколько дочерних элементов. Задача панели — это определение местоположения дочерних элементов.

Видов панелей несколько — от простых, который позволяют позиционировать элементы точно по их координатам, или выстраивать в горизонтальные или вертикальные ряды, до более сложных, которые позволяют компоновать объекты в виде сетки.

Документы нефиксированного формата

WPF позволяет оформлять документы, которые подстраивают своё содержимое под доступное им пространство, с учётом заданных правил форматирования.

Flow documents are designed to optimize viewing and readability. Rather than being set to one predefined layout, flow documents dynamically adjust and reflow their content based on run-time variables such as window size, device resolution, and optional user preferences. In addition, flow documents offer advanced document features, such as pagination and columns. This topic provides an overview of flow documents and how to create them.



A UI Elements may be embedded directly in flow content

A flow document is designed to "reflow content" depending on window size, device resolution, and other environment variables. In addition, flow documents have a number of built in features including search, viewing modes that optimize readability, and the ability to change the size and

appearance of fonts. Flow Documents are best utilized when ease of reading is the primary document consumption scenario. In contrast, Fixed Documents are designed to have a static presentation. Fixed Documents are useful when fidelity of the source content is essential. See Documents in WPF for more information on different types of documents.

Можно интегрировать текст в пользовательский интерфейс, а можно интегрировать пользовательский интерфейс в текст.

WPF предоставляет возможности обработки текста, сопоставимые с HTML. Можно использовать разные виды форматирования текста, таблицы, списки, нумерованные и ненумерованные, и так далее. Также есть возможность разбить содержимое на страницы и колонки, а также адаптировать текст под доступную рабочую область.

Обзор XAML

XAML позволяет описать элементы пользовательского интерфейса с помощью языка разметки, поместив описание поведения в отдельные файлы с кодом.

Такое отделение разметки от поведения называется code-behind.

Чувствителен к регистру.

Alice != alice != aLiCe

Элемент

Предназначены для описания экземпляров классов. После того, как разметка будет обработана XAML-анализатором, будут созданы объекты соответствующих типов.

Для их создания применяется конструктор по умолчанию.

`<TypeName>Content</TypeName>`

Описание состоит из **открывающего тэга**, **закрывающего тэга** и **содержимого**.

TypeName – это имя элемента, которое является по сути названием класса.

После имени могут располагаться атрибуты.

Содержимым может быть обычный текст, или другие элементы.

Обычно содержимое строго описано типом данных элемента.

Если элемент не имеет содержимого, то можно ничего не писать между тэгами, или же записать его в виде самозакрывающегося тэга:

```
<TypeName></TypeName>
```

```
<TypeName/>
```

Атрибут

```
<TypeName PropertyName="Value">Content</TypeName>
```

```
<TypeName PropertyName="Value"/>
```

```
<TypeName PropertyName1="Value1" PropertyName2="Value2"/>
```

```
<Button Height="30" Width="100">Yes</Button>
```

```
var button = new Button { Content = "Yes", Height = 30.0, Width = 100.0 };
```

```
<TypeName EventName="HandlerName"/>
```

```
<Button Click="Button_Click">OK</Button>
```

Атрибут нельзя описывать более одного раза.

```
<Button Height="100" Height="50">OK</Button>
```