БАЗОВЫЕ СВОЙСТВА ЭЛЕМЕНТОВ

Разбирая объектную модель `WPF` можно предположить, что у каждого элемента есть общие свойства, так как он наследует их от базовых классов. Это действительно так. `WPF` нацелен на гибкость и учитывает то, что элемент находится в интерфейсе. Интерфейсу необходима некоторая информация для того, чтобы элемент мог занять правильную позицию, а также доступен для модификации.

Свойства классов

Каждый элемент в конечном счёте превращается в экземпляр класса. Свойства имени указываются автоматически, однако автоматическое имя нам недоступно.

Все классовые свойства предоставляются из пространства имён `xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"`.

Технология `IntelliSense` при обращении к псевдониму пространства имён x отображает возможные атрибуты.

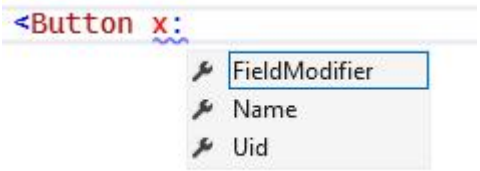

Рисунок 1

Имя экземпляра

Чтобы указать конкретное имя, по которому к экземпляру можно обратиться, используют свойство `x:Name`.

```
<Button x:Name="MyButton"/>
```

Теперь в связанном классе можно обращаться по имени `MyButton` как к полю экземпляра класса `Button`.

```
public MainWindow()
{
    InitializeComponent();

    MyButton.Content = "Это значение присвоено из C#";
}
```

Рисунок 2

Указание на модификатор доступа

Изначально все поля класса объявляются как приватные. В случае, когда общий элемент *(в нашем примере это **Window**)* передаётся в управление другому объекту, при необходимости можно изменить модификатор доступа. В кавычках указывается модификатор, подобный модификатору языка **C#**.

```
<Button x:Name="MyButton"
        x:FieldModifier="public"/>
```

Рисунок 3

Указание уникального идентификатора

Используется, когда в разметке необходимо указать на конкретный элемент. Задаётся в виде строки.

```
<Button x:Name="MyButton"
        x:FieldModifier="public"
        x:Uid="BID123"/>
```

Рисунок 4

Указатель ключа ресурса

Нередко возникают ситуации, когда в **XAML**-разметке описываются ресурсы. Элемент Button не является ресурсом, однако кисть, или рисунки вполне подойдут на эту роль. Для них можно и порой нужно указать атрибут **x:Key**. Они используется только для идентификации ресурсов.

```
<Window.Resources>
    <SolidColorBrush x:Key="MyBrush" Color="Aqua"/>
</Window.Resources>
```

Рисунок 5

Атрибуты размеров

Значения атрибутов, отвечающих за размеры, задаются вещественным числом *(как с дробной частью, так и без)*.

Если у элемента не указаны размеры явно, они установятся автоматически, исходя из контейнера, в котором он находится.

Базовая единица измерения - **px**. Эта единица обозначает **1/96 дюйма** и является независимой от разрешения экрана.

```
<Button Width="24"/>
```

Рисунок 6

Однако вы можете явно указать единицу измерения, указав после значения приставку.

- **px** (по умолчанию) — это не зависящие от устройства единицы измерения (1/96-й дюйм на единицу)
- **in** - дюймы; 1in===96 пикселей
- **cm** — сантиметры; 1cm==(96/2.54) px *(в одном дюйме 2.54 см)*
- **pt** является точками; 1pt==(96/72) px
- **Auto** – относительно содержимого

```
<Button Width="5cm"/>
```

Рисунок 7

● Ширина

Ширина элемента задаётся свойством **Width**. Значение оформляется по правилам указания размеров.

```
<Button Width="50"/>
```

Рисунок 8

● Высота

Высота элемента задаётся свойством **Height**. Значение оформляется по правилам указания размеров.

```
<Button Width="50"
        Height="50"/>
```

Рисунок 9

● Максимальная ширина

Из-за динамичного интерфейса без указания размеров элемента они автоматически подстраиваются относительно контейнера, в котором находятся. Иногда требуется ограничить динамичность размеров. Для этого в **WPF** существуют специальные свойства.

Для того, чтобы указать максимально возможную ширину используют атрибут **MaxWidth**. Значение оформляется по правилам указания размеров.

```
<Button MaxWidth="70"/>
```

Рисунок 10

● Минимальная ширина

Для того, чтобы указать минимально возможную ширину используют атрибут **MinWidth**. Значение оформляется по правилам указания размеров.

```
<Button MinWidth="70"/>
```

Рисунок 11

● Максимальная высота

Для того, чтобы указать максимально возможную высоту используют атрибут **MaxHeight**. Значение оформляется по правилам указания размеров.

```
<Button MaxHeight="70"/>
```

Рисунок 12

● Минимальная высота

Для того, чтобы указать минимально возможную высоту используют атрибут **MinHeight**. Значение оформляется по правилам указания размеров.

```
<Button MinHeight="70"/>
```

Рисунок 13

● Отступы

Элемент может иметь отступы от границ контейнера, в котором находится. Для указания отступа используют атрибут **`Margin`**. Значение оформляется по правилам указания размеров.

```
<Button Margin="15"/>
```

Рисунок 14

Если необходимо указать отступ конкретной стороны элемента, необходимо указать значение отступа для каждой стороны. Для этого в значении необходимо указать перечисление. В **`WPF`** для свойства Margin принят отсчёт от левой стороны по часовой стрелке: слева, сверху, справа, снизу. Вообще, позиционирование всегда начинается где-то слева сверху. Это правило удобно, так как человек читает текст слева направо и сверху вниз.

```
<Button Margin="15 0 0 0"/>
```

Рисунок 15

Для указания одинаковых отступов для противоположных сторон так же имеется сокращённая запись. Тогда в записи используют два числа. Первое указывает на размер отступа для левой и правой стороны, а второе для верхней и нижней.

```
<Button Margin="15 0"/>
```

Рисунок 16

Выравнивание

Выравнивание всегда указывается относительно измерения. Для двумерного изображения выделяют два измерения: горизонтальное и вертикальное.

Значения для указания выравнивания по **горизонтали**:

- **`Left`** - по левому краю
- **`Center`** - по центру

- `Right` - по правому краю
- `Stretch` - растянуть по горизонтали *(по ширине).*

Значения для указания выравнивания по **вертикали**:

- `Top` - выравнивание по верхней границе
- `Center` - выравнивание по центру
- `Bottom` - выравнивание по нижней границе
- `Stretch` - растянуть по вертикали *(по высоте)*

Выравнивание по горизонтали

Для указания выравнивания по горизонтали устанавливают атрибут `HorizontalAligment`. Значение берут из перечисления.

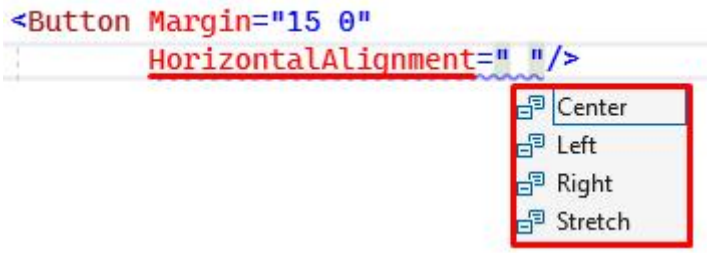

Рисунок 17

Выравнивание по вертикали

Для выравнивания по вертикали устанавливают атрибут `VerticalAligment`. Значение берут из перечисления.

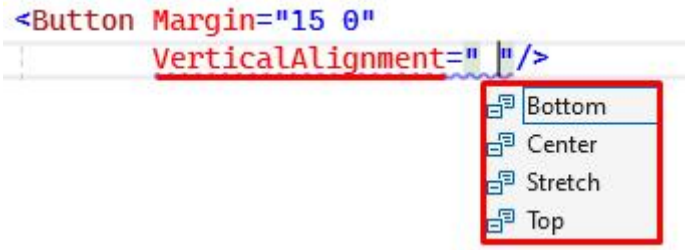

Рисунок 18

Выравнивание содержимого

Для элементов, которые могут иметь содержимое *(свойство **Content**)* можно так же указать его выравнивание. Это делается с помощью атрибутов **HorizontalContentAligment** для выравнивания содержимого по горизонтали и **VerticalContentAligment** для выравнивания содержимого по вертикали. Значения задаются из перечисления.

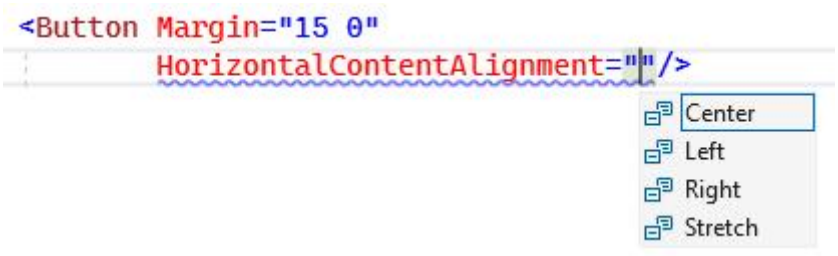

Рисунок 19

Сортировка слоёв

Каждый элемент, добавляемый в дерево разметки, в каком-то смысле образует слой. Его иногда называют третьим измерением. В том плане, что каждый последующий элемент накладывается на предыдущий. Для того, чтобы указать порядок вручную вне зависимости от автоматического расположения, используют свойство **Panel.ZIndex**. Чем больше число - тем выше элемент по слою. На переднем плане всегда будет оказываться элемент с максимальным значением этого атрибута. Значение указывается в виде целого положительного числа. Класс **Panel** - это базовый класс для элементов, участвующих в компоновке.

```
<Button Margin="15 0"
        Panel.ZIndex="2"/>
```

Рисунок 20

Отображение элемента

Отображением элемента удобно управлять, когда оно устанавливается автоматически. Это может происходить, например, относительно статуса пользователя: кому-то кнопку стоит отображать, кому-то нет.

Для определения значения отображения элемента используют атрибут **`Visibility`**. Его значение выбирается из перечисления:

- **`Visible`** (по умолчанию) - отображать
- **`Collapsed`** - скрыть полностью
- **`Hidden`** - скрыть элемент, но оставить место под него в разметке *(так, как будто элемент есть, но сам не отображается)*.

Получение фокуса

Обычно, элементы реагируют на клик мыши, получая на себя фокус. Если необходимо «скрыть» элемент от получения фокуса, необходимо установить значение атрибута **`IsHitTestVisible`** равным **`false`**. По умолчанию - **`true`**.

```
<TextBlock Text="NoMouse!"
           IsHitTestVisible="False"/>
```

Рисунок 21