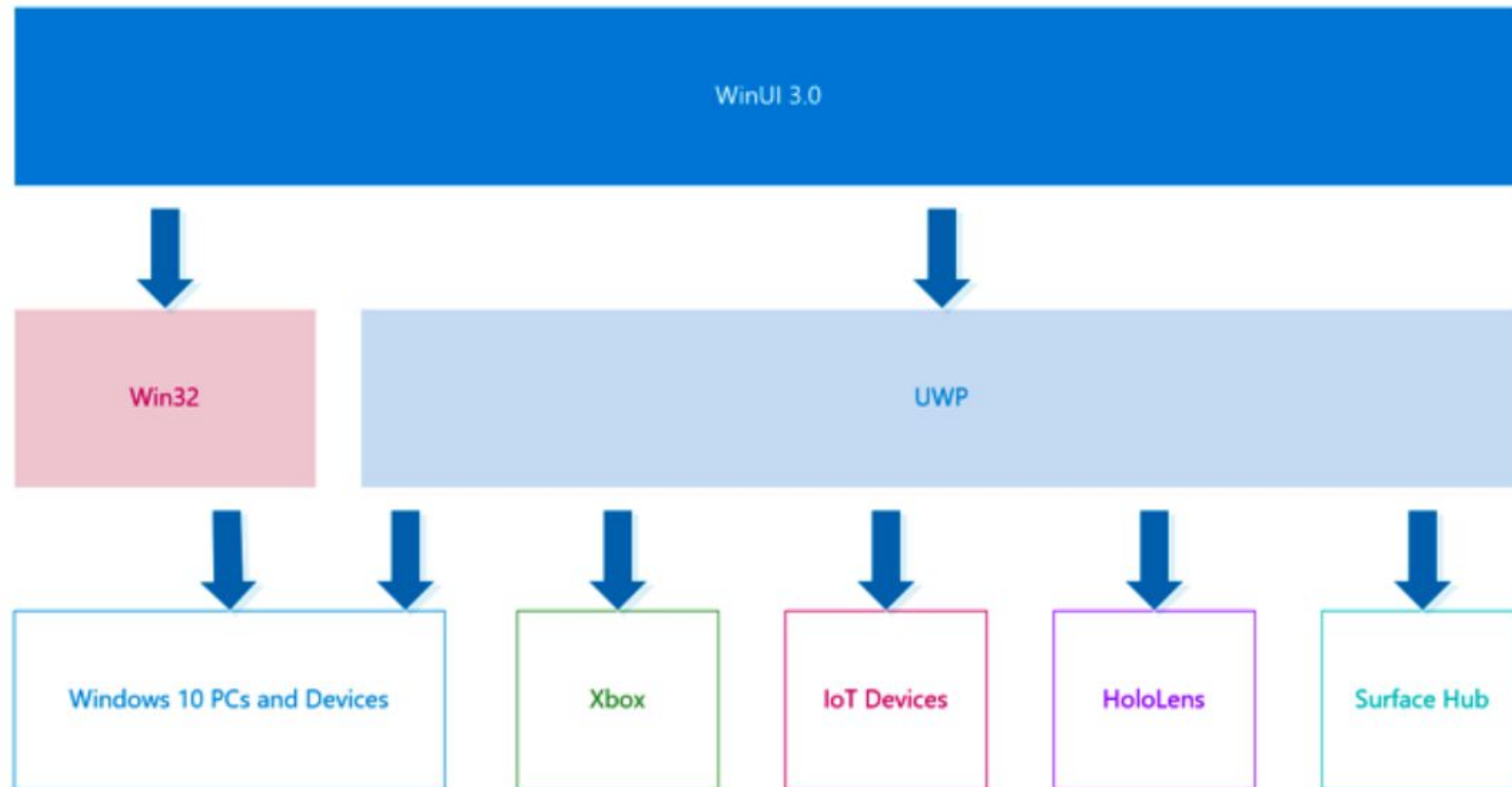


# WinUI



The modern native UI platform of  
Windows.

WinUI позволяет быстро создавать приложения Windows, стиль которых адаптируется к целевой платформе



# Windows Community Toolkit - комплект инструментов с открытым исходным кодом содержащий десятки элементов управления и других вспомогательных библиотек для WinUI, UWP и Win32

About - Windows Community Toolkit Sample App

← Controls Animations Services Helpers Brushes Extensions Gaze

## Get Started

The Windows Community Toolkit is a collection of helpers, extensions, and custom controls. It simplifies and demonstrates common developer tasks for building UWP and .NET apps for Windows 10.

### Recent Activity

<a href="#">Win2d Path Mini Language Parser</a>	<a href="#">ColorPicker</a>
<a href="#">Microsoft Graph Service</a>	<a href="#">Visual Extensions</a>
<a href="#">VisualEffectFactory</a>	<a href="#">AcrylicBrush</a>
<a href="#">Toast</a>	<a href="#">RangeSelector</a>
<a href="#">MarkdownTextBlock</a>	<a href="#">Fade</a>

### What's New

- [MVVM Toolkit](#)
- [EffectAnimations](#)
- [TabbedCommandBar](#)
- [ColorPicker](#)
- [ColorPickerButton](#)
- [SwitchPresenter](#)

### ToolKit

- [GitHub Repository](#)
- [Documentation](#)
- [GitHub Issues](#)

### Useful links

- [Developer Center](#)
- [Get started with Windows apps](#)
- [Windows apps design](#)

### Release Notes

- [Version 7.0.0-preview5 notes](#) Feb 24
- [Version 7.0.0-preview4 notes](#) Nov 12

# MVVM



**WinUI** – это набор элементов управления и библиотек с открытым исходным кодом, предназначенных для использования в приложениях UWP и Win32.

# Сравнение приложений для Windows 8 и Windows 10

10	Приложения Windows и XAML	Приложения Windows 10 UWP
Тип окна	Только полноэкранное	С изменяемым размером
Тип устройства	Работает только на ПК	Различные устройства под управлением Windows 10
Количество экземпляров	1	1 (по умолчанию) или несколько
Поддержка консольных приложений	Нет	Да
Доступ к файловой системе	В песочнице – только локальное хранилище	По умолчанию в песочнице Приложение может запросить дополнительный доступ к пользовательским

# Metro style



Основное преимущество WinUI по сравнению с UWP заключается в **меньшей зависимости приложений Windows от конкретной версии Windows.**

Элементы управления, стили и API вынесены за пределы Windows SDK и сопровождаются отдельно



# XAML

основан на расширяемом языке разметки (XML)

WinUI.SimpleSample

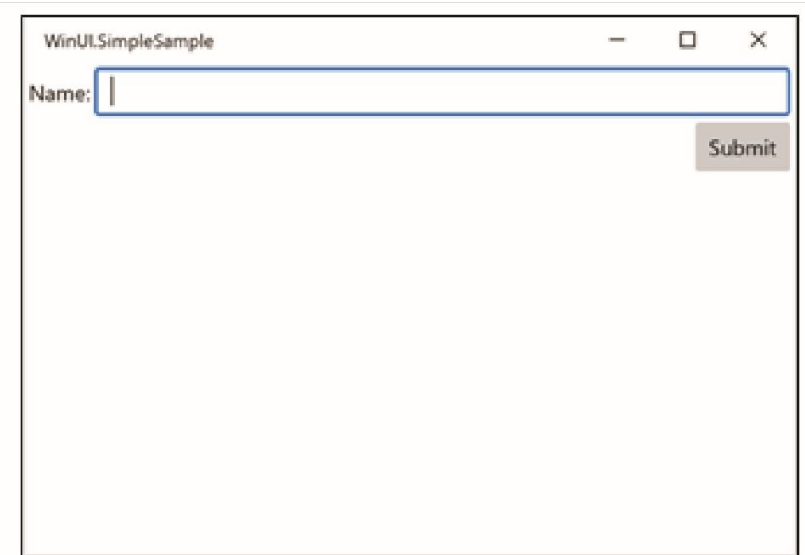
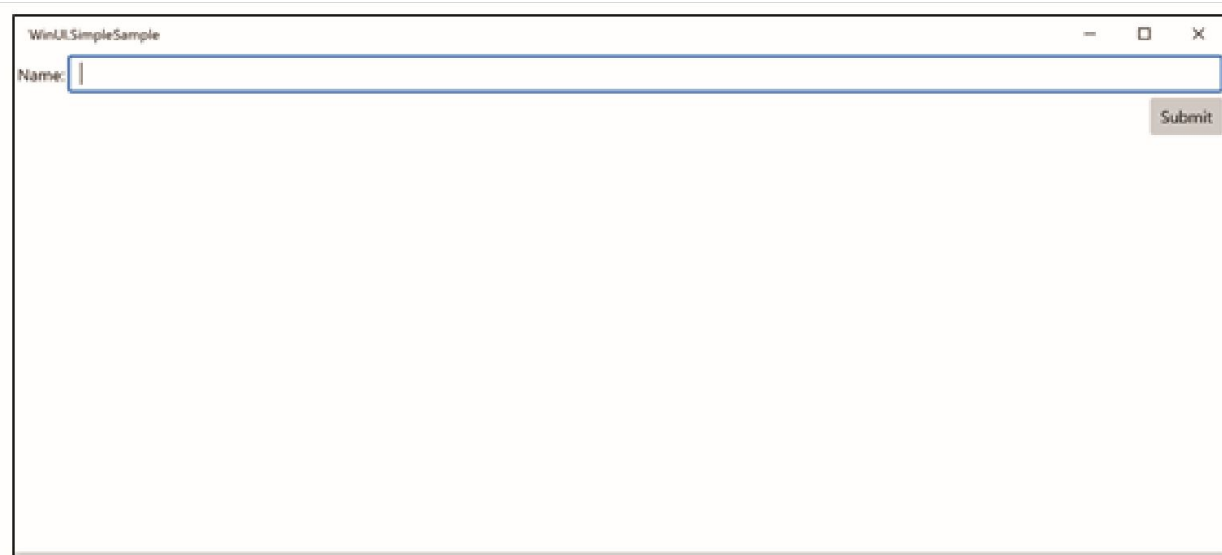
Name: |

Submit

```
<Grid Width="400" Height="250" Padding="2"
  HorizontalAlignment="Center"
  VerticalAlignment="Center">
  <Grid.RowDefinitions>
    <RowDefinition Height="Auto"/>
    <RowDefinition Height="*/>
  </Grid.RowDefinitions>
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="Auto"/>
    <ColumnDefinition Width="*/>
  </Grid.ColumnDefinitions>
  <TextBlock Grid.Row="0" Grid.Column="0"
    Text="Name:"
    Margin="0,0,2,0"
    VerticalAlignment="Center"/>
  <TextBox Grid.Row="0" Grid.Column="1"
    Text=""/>
  <Button Grid.Row="1" Grid.Column="1" Margin="0,4,0,0"
    HorizontalAlignment="Right"
    VerticalAlignment="Top"
    Content="Submit"/>
</Grid>
```

# Адаптивный интерфейс

```
<Grid VerticalAlignment="Top" HorizontalAlignment="Stretch"  
Padding="2">
```



# данные

e

## 1. Создание класса MainViewModel

интерфейс `INotifyPropertyChanged` - нужен для получения пользовательским интерфейсом событий обновления при изменении привязанных к данным свойств

```
public class MainViewModel : INotifyPropertyChanged
{
    public event PropertyChangedEventHandler PropertyChanged;
    private string _name;

    public MainViewModel()
    {
        Name = "Bob Jones";
    }

    public string Name
    {
        get
        {
            return _name;
        }
        set
        {
            if (_name == value) return;

            _name = value;
            PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(nameof(Name)));
        }
    }
}
```

## 2. Создание экземпляра класса и его задание в качестве свойства на MainPage

```
public sealed partial class MainPage : Page
{
    public MainPage()
    {
        this.InitializeComponent();

        this.ViewModel = new MainViewModel();
    }

    public MainViewModel ViewModel { get; set; }
}
```

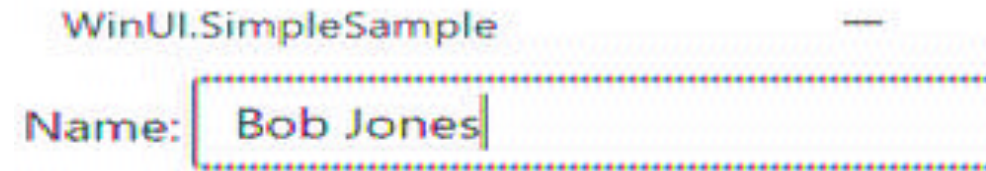
### 3. Добавление кода привязки к данным в XAML разметку элемента TextBox

```
<TextBox Grid.Row="0" Grid.Column="1" Text="{x:Bind  
Path=ViewModel.Name, Mode=TwoWay}"/>
```

Путь  
привязки

Режим двусторонней  
привязки

Расширение  
разметки



# Стилизация UI с помощью

## XAML

XAML поддерживает применение стилей в любой области видимости:

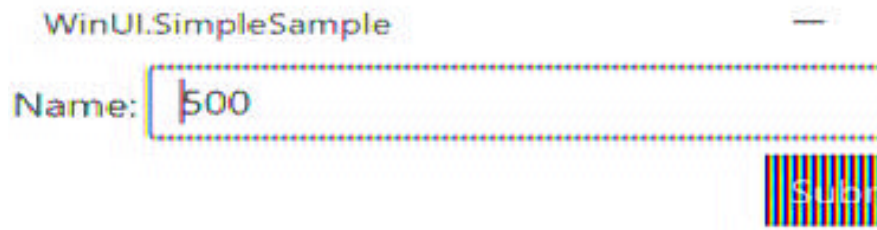
- Глобально (файл app.xaml)
- На текущей странице (в объявлении Page.Resources)
- Внутри любого уровня или вложенного элемента на странице

# Пример использования свойства style

1. Перемещение кнопки Submit в StackPanel и добавление кнопки Cancel

```
<StackPanel Grid.Row="1" Grid.Column="1"  
    Margin="0,4,0,0"  
    HorizontalAlignment="Right"  
    VerticalAlignment="Top"  
    Orientation="Horizontal">  
    <Button Content="Submit" Margin="0,0,4,0"/>  
    <Button Content="Cancel"/>  
</StackPanel>
```

## 2. Добавление блока Style в секцию Page.Resources для стилизации кнопок.



```
<Page.Resources>  
  <Style TargetType="Button">  
    <Setter Property="BorderThickness"  
      Value="2" />  
    <Setter Property="Foreground"  
      Value="LightGray" />  
    <Setter Property="BorderBrush"  
      Value="GhostWhite"/>  
    <Setter Property="Background"  
      Value="DarkBlue" />  
  </Style>  
</Page.Resources>
```