.NET MAUI All-in-One Templates Pack

Contents

Introduction

This VS extension is loaded with Projects, Item Templates, and Code Snippets for working with .NET MAUI in Visual Studio 2022 v17.6.0 and later, and Visual Studio 2026 Insiders v18.0.0 (from extension v7.9.0.0) (on both x64 and ARM64 installations).

Project Templates

- .NET MAUI App – All All-in-One App Project Template. For more details, check out these articles
- .NET MAUI Class Library
- Shared Class Library (targeting both Xamarin.Forms and .NET MAUI)

Item Templates

Made available in the section titled **MAUI** in the **Add New Item** dialog.

Generic Template in XAML and C# and has been named as:

- Generic Item
- Generic Item (C#)

ContentPage in XAML, C#, and Razor and has been named as:

- Content Page
- Content Page (C#)
- Content Page (Razor)
- Content Page with BlazorWebView
- Content Page with BlazorWebView (C#)
- Content Page with ViewModel
- Content Page (C#) with ViewModel

ContentView in XAML, C#, and Razor and has been named as:

- Content View
- Content View (C#)
- Content View (Razor)

Shell, in XAML, C#, and Razor, is a page for defining app visual hierarchy along with navigation.

ResourceDictionary, a page for managing resources made available with C# code-behind file and XAML only (as the code-behind is used rarely).

Templates for creating a Custom View definition:

- Custom View and Handler (Regular) (.NET MAUI)
- Custom View and Handler (Cond.) (.NET MAUI)
- Custom View and Renderer (Regular) (.NET MAUI)
- Custom View and Renderer (Cond.) (.NET MAUI)

- **Regular type template** generates the Handler / Renderer source files in the Platforms folder whereas **Cond. type template** houses all of them in a single folder.
- *For conditional type format, ensure Conditional Compilation is configured in the project file for the build to succeed. An additional option is provided during project creation (in both VS IDE and CLI) (or manually thereafter). Check out this* ***readme*** *for further details.*

Partial Class, a C# class (partial), useful for defining *ViewModel type* with the *MVVM Toolkit*, is made available in the section titled **Code**.

Code Snippets

For XAML

On the XAML page, type the short name and hit the Tab key twice to insert the snippet.

Snippets mentioned in boldface also work as a **SurroundWith** snippet too (from the **Xaml** section).

In the Output Format column, text highlighted in different colors infers the following:

- Yellow colors are placeholders where the user can modify the values
- Green colors are derived values and can't be modified. For example, containing the class name

- Turquoise colors are reflected values, where the placeholder value is filled in

Snippet	Short Name	Output Format
Grid	**grid1**	<Grid ColumnDefinitions="" RowDefinitions=""> </Grid>
Flex Layout	**flex**	<FlexLayout> </FlexLayout>
Stack Layout	**stack**	<StackLayout> </StackLayout>
Horizontal Stack Layout	**hstack**	<HorizontalStackLayout> </HorizontalStackLayout>
Vertical Stack Layout	**vstack**	<VerticalStackLayout> </VerticalStackLayout>
Style	**style**	<Style TargetType="Page"> <Style>
Color	color	<Color x:Key="Success">Green</Color>
Resources	res	<ContentPage.Resources> <ResourceDictionary> </ResourceDictionary> </ContentPage.Resources>
Gestures	gesture	<Label.GestureRecognizers> </Label.GestureRecognizers>
Tap Gesture Recognizer	tap	<TapGestureRecognizer />
Drag Gesture Recognizer	drag	<DragGestureRecognizer />
Drop Gesture Recognizer	drop	<DropGestureRecognizer />
Pan Gesture Recognizer	pan	<PanGestureRecognizer />
Pinch Gesture Recognizer	pinch	<PinchGestureRecognizer />

Snippet	Short Name	Output Format
Pointer Gesture Recognizer	Pointer	<PointerGestureRecognizer />
Swipe Gesture Recognizer	swipe	<SwipeGestureRecognizer />
Blazor Web View	bwv	<BlazorWebView HostPage="wwwroot/index.html"> <BlazorWebView.RootComponents> <RootComponent ComponentType="{x:Type }" Selector="#app" /> </BlazorWebView.RootComponents> </BlazorWebView>
.NET MAUI Blazor Namespace	mb	xmlns:b="clr-namespace:Microsoft.AspNetCore.Components.WebView.Maui ;assembly=Microsoft.AspNetCore.Components.WebView.Maui"
WPF Blazor Namespace	wb	xmlns:b="clr-namespace:Microsoft.AspNetCore.Components.WebView.Wpf ;assembly=Microsoft.AspNetCore.Components.WebView.Wpf"

For C#

In the C# code file, type the short name and hit the Tab key twice to insert the snippet.

Snippets mentioned in boldface also work as a **SurroundWith** snippet (from the **CSharp** section).

In the Output Format column, text highlighted in different colors infers the following:

- Yellow colors are placeholders where the user can modify the values
- Green colors are derived values, and can't be modified. For example, containing the class name
- Turquoise colors are reflected values, where the placeholder value is filled in

Snippet	Short Name	Output Format

Async Event Handler	**aeh**	private async void MyMethod(object sender, EventArgs e)
		
		{
		
		}

Attached Property	propap	*Here assuming MyClass is the containing type.*
		
		public static readonly BindableProperty NameProperty = BindableProperty.CreateAttached(nameof(NameProperty), typeof(string), typeof(MyClass), default(string));
		
		public static string GetName(BindableObject bindable) => (string)bindable.GetValue(NameProperty);
		
		public static void SetName(BindableObject bindable, string value) => bindable.SetValue(NameProperty, value);

Bindable Property	propbp	*Here assuming MyClass is the containing type.*
		
		public static readonly BindableProperty NameProperty = BindableProperty.Create(nameof(Name), typeof(string), typeof(MyClass), default(string));
		
		public string Name
		
		{
		
		get => (string)GetValue(NameProperty);
		
		set => SetValue(NameProperty, value);
		
		}

Comet Property (MVU)	propc
	
	(This has been shortened to **propc** from **propcomet**)
		public string Name
		
		{
		
		get => GetProperty<string>();
		
		set => SetProperty(value);
		
		}

Snippet	Short Name	Output Format
Cross Platform	cp (This has been updated to **cp** from **xplat**)	#if ANDROID #elif IOS #elif MACCATALYST #elif TIZEN #elif WINDOWS #endif

Event Handler	**eh**	private void MyMethod(object sender, EventArgs e)
		
		{
		
		}

Method	**method**	private void MyMethod()
		
		{
		
		}

Async Method	**amethod**	private async Task MyMethod()
		
		{
		
		}

Record
 	
(C# 9.0 or higher)
	record	record MyRecord
		
		{
		
		}

Record Struct
 	
(C# 10.0 or higher)
	rstruct
	
	(This has been updated to **rstruct** from **recstruct**)
		record struct MyRecStruct
		
		{
		
		}

Observable Property *(CommunityToolkit.Mvvm)*	propop	[ObservableProperty] private string name;

Snippet	Short Name	Output Format

Relay Command
 	
(CommunityToolkit.Mvvm)
	rcmd	[RelayCommand]
		
		private void DoSomething()
		
		{
		
		}

Async Relay Command
 	
(CommunityToolkit.Mvvm)
	arcmd	[RelayCommand]
		
		private async Task DoSomethingAsync()
		
		{
		
		}

ViewModel Property	propvm	private string name;
		
		public string Name
		
		{
		
		get => name;
		
		set => SetProperty(ref name, value);
		
		}

C# Markup Extension Method	cmem	public static TBindable MyMethod<TBindable>(this TBindable bindable) where TBindable : BindableObject
		
		{
		
		  return bindable;
		
		}

Support

This VS extension works with Visual Studio 2022 v17.6.0 and later, and Visual Studio 2026 Insiders v18.0.0, as long as the .NET MAUI workload is installed. It supports both Stable (.NET MAUI 9) releases and Previews (.NET MAUI 10) of .NET MAUI on x64 and ARM64 VS installations. To stay up-to-date with future updates in upcoming .NET MAUI versions, an extension update will be provided as necessary.

If you encounter any issues or have suggestions for improving these templates, please log them as issues here.