.NET MAUI Project, Item Templates and Code Snippets

This VS extension is loaded with Project, Item Templates and Code Snippets for working with .NET MAUI in Visual Studio 2022.

This has project template for MAUI, MAUI Blazor and MAUI Class Library project and it has been named as:

- .NET MAUI App (Preview 13) – All All-in-One App Project Template. For more details check out this blog post
- .NET MAUI App (C#) (Preview 13)
- .NET MAUI Class Library (Preview 13)

And reg. Item templates (find them in the section titled **MAUI**):

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

- Content Page (.NET MAUI)
- Content Page (C#) (.NET MAUI)

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

- Content View (.NET MAUI)
- Content View (C#) (.NET MAUI)

A page for defining app visual hierarchy with Shell and a page for managing resources with Resource Dictionary, both in XAML.

A XAML only Resource Dictionary template has been added as the C# code-behind is used rarely.

Has XAML Code Snippets for:

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

Snippets mentioned in bold-face also works as a **SurroundWith** snippet too (from **Xaml** section).

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

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

Snippet	Short Name	Output Format
Grid	**grid**	<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
Swipe Gesture Recognizer	swipe	<SwipeGestureRecognizer />
Blazor Web View	bwv	<b:BlazorWebView HostPage="wwwroot/index.html"> <b:BlazorWebView.RootComponents> <b:RootComponent ComponentType="{x:Type }" Selector="#app" /> </b:BlazorWebView.RootComponents> </b:BlazorWebView>

Has C# Code Snippets for:

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

Snippets mentioned in bold-face also works as a **SurroundWith** snippet too (from **CSharp** section).

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

- Yellow color are placeholders where user can modify the values
- Green color are derived values, can't be modified. For example, containing class name
- Torquoise color 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);

Snippet	Short Name	Output Format

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);
		
		}

Cross Platform	cp (This has been updated to **cp** from **xplat**)	#if ANDROID #elif IOS #elif MACCATALYST #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() {

Snippet	Short Name	Output Format
		}

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
		{
		}

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;
		}

Note: Currently, these project templates target .NET MAUI Preview 13 (VS2022 17.2 Preview 1.0 or later) and to support further changes in newer .NET MAUI releases, an update to this VS extension will be made available accordingly.