tutorial

Windows App SDK

Buy me a coffee

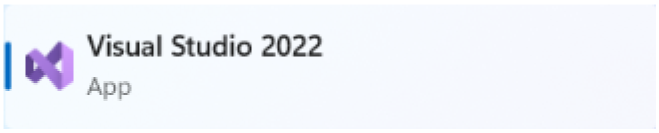
Badge Notifications

Badge Notifications shows how you can use **`BadgeNotification`** with the **Windows App SDK**. This allows you to display different **Badges** on the **Icon** for your Application in the **Taskbar** of **Windows**.

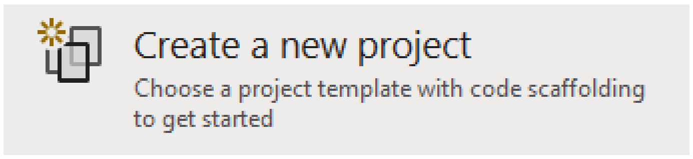
Step 1

Follow **Setup and Start** on how to get **Setup** and **Install** what you need for **Visual Studio 2022** and **Windows App SDK**.

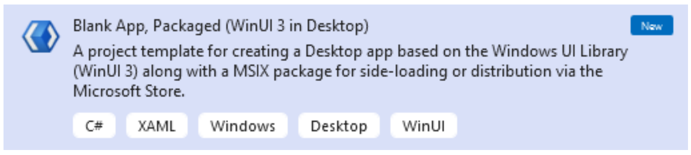
In **Windows 11** choose **Start** and then find or search for **Visual Studio 2022** and then select it.



Once **Visual Studio 2022** has started select **Create a new project**.

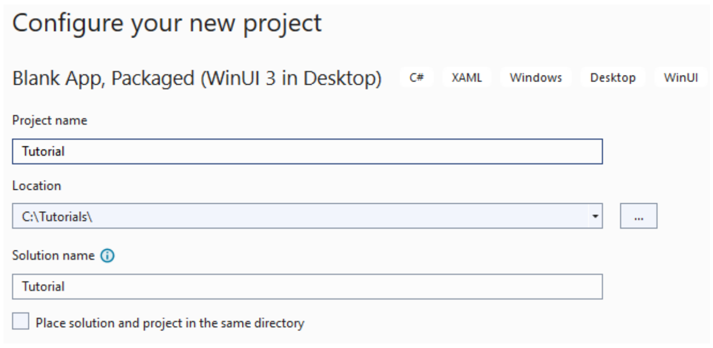

Create a new project
Choose a project template with code scaffolding to get started

Then choose the **Blank App, Packages (WinUI in Desktop)** and then select **Next**.


Blank App, Packaged (WinUI 3 in Desktop)

A project template for creating a Desktop app based on the Windows UI Library (WinUI 3) along with a MSIX package for side-loading or distribution via the Microsoft Store.

After that in **Configure your new project** type in the **Project name** as *BadgeNotifications*, then select a Location and then select **Create** to start a new **Solution**.


Configure your new project

Blank App, Packaged (WinUI 3 in Desktop)

Project name: Tutorial

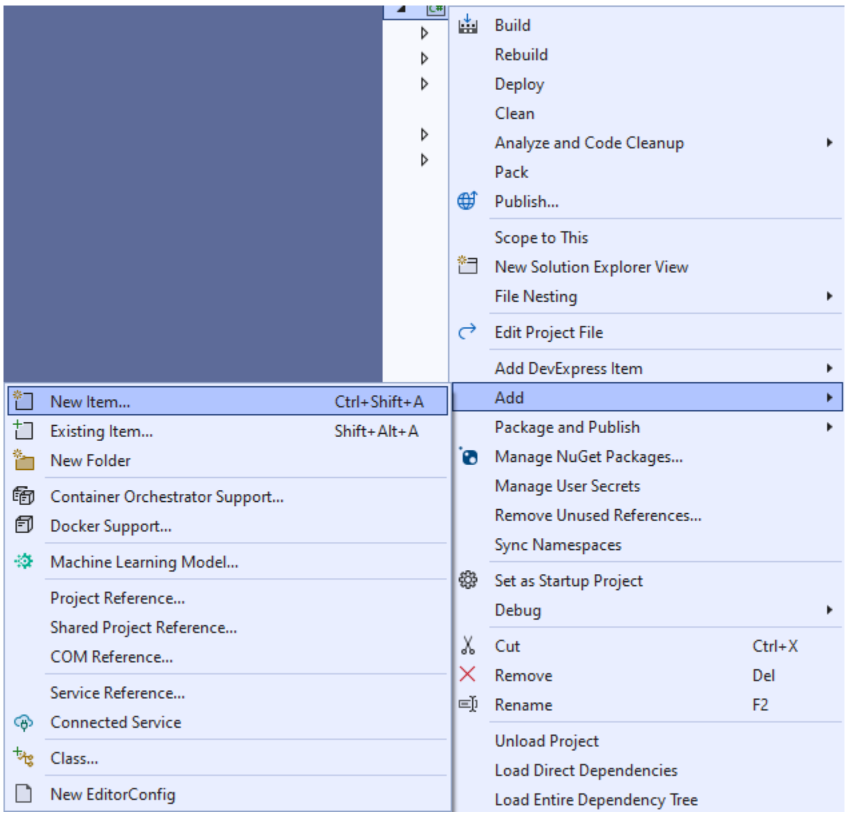
Location: C:\Tutorials\

Solution name: Tutorial

Place solution and project in the same directory

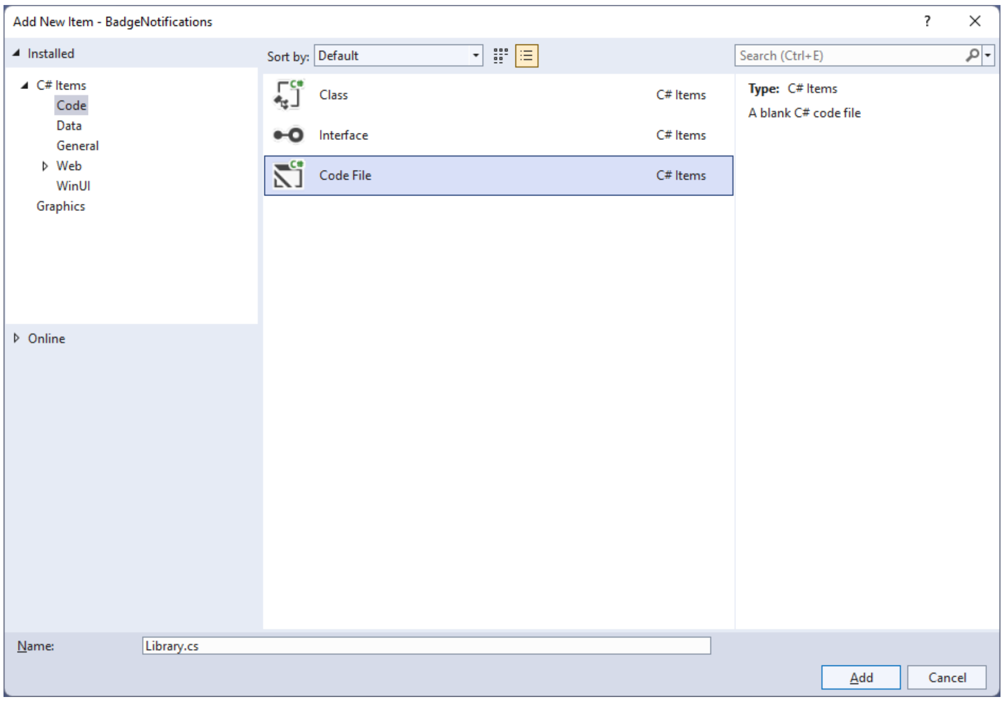
Step 2

Then in **Visual Studio** within **Solution Explorer** for the **Solution**, right click on the **Project** shown below the **Solution** and then select **Add** then **New Item...**



Step 3

Then in **Add New Item** from the **C# Items** list, select **Code** and then select **Code File** from the list next to this, then type in the name of *Library.cs* and then **Click** on **Add**.



Type: C# Items
A blank C# code file

Name: Library.cs

Step 4

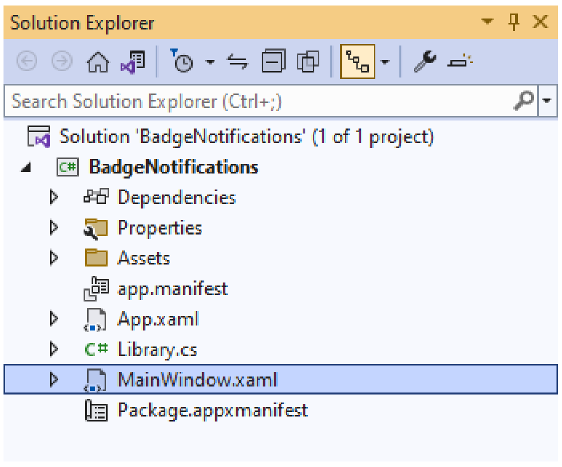
You will now be in the **View** for the **Code** of *Library.cs*, within this type the following **Code**:

```
using Microsoft.UI.Xaml.Controls;
using System.Collections.Generic;
using Windows.Data.Xml.Dom;
using Windows.UI.Notifications;

internal class Library
{
    public List<string> Options => new()
    {
        "number", "activity", "alarm", "attention", "available", "away",
        "busy", "error", "newMessage", "paused", "playing", "unavailable"
    };

    public void SetBadge(ComboBox options, TextBox number)
    {
        var selected = options.SelectedValue as string;
        var result = selected == "number" ? number.Text : selected;
        XmlDocument badge = BadgeUpdateManager.GetTemplateContent(
        int.TryParse(result, out _) ?
            BadgeTemplateType.BadgeNumber :
            BadgeTemplateType.BadgeGlyph);
        XmlNodeList attributes = badge.GetElementsByTagName("badge");
        attributes[0].Attributes.GetNamedItem("value").NodeValue = result;
        BadgeNotification notification = new(badge);
        BadgeUpdateManager.CreateBadgeUpdaterForApplication().Update(notification);
    }

    public void ClearBadge()
    {
        BadgeUpdateManager.CreateBadgeUpdaterForApplication().Clear();
    }
}
```

The **`Class`** that has been defined in *Library.cs* has a **Property** for **`Options`**, with the exception **`number`** these are all the different types of **Glyph** that can be shown for a **`BadgeNotification`**. Then there is a **Method** of **`SetBadge`** which will get the **`SelectedValue`** of a **`ComboBox`** passed in. To get the **Value** to use, if the selected option was **`number`** then it should use the contents of a **`TextBox`** passed in, which is the **Property** for **`Text`** along with the **`BadgeTemplateType.BadgeNumber`** or it should use the one for **`BadgeTemplateType.BadgeGlyph`**. The lines to do this use a value that is a **`bool`** before **`?`** which if **`true`**, the value after **`?`** will be used, if **`false`** the value after **`:`** will be used, these together are **Conditional Operators**. There is some code to build up the elements of the **Badge Notification** using **XML** which is needed to create the **`BadgeNotification`** and then this is used with the **`BadgeUpdateManager.`** The other **Method** is used to Clear the **`BadgeNotification`** using with the **`BadgeUpdateManager`**.

Step 5

Then from **Solution Explorer** for the **Solution** double-click on **MainWindow.xaml** to see the **XAML** for the **Main Window**.



Step 6

In the **XAML** for **MainWindow.xaml** there be some **XAML** for a **`StackPanel`**, this should be **Removed** by removing the following:

```
<StackPanel Orientation="Horizontal"
HorizontalAlignment="Center" VerticalAlignment="Center">
    <Button x:Name="myButton" Click="myButton_Click">Click Me</Button>
</StackPanel>
```

Step 7

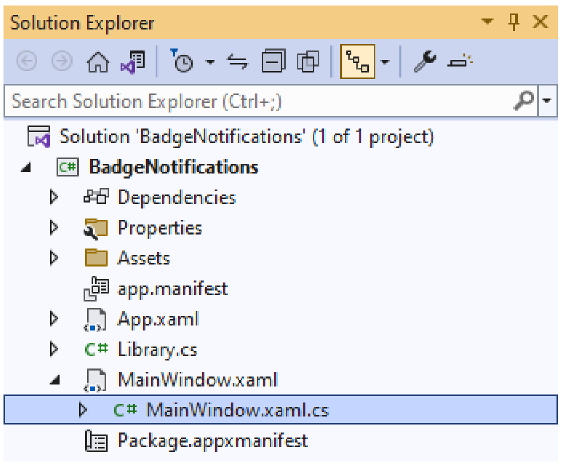
While still in the **XAML** for **MainWindow.xaml** above **`</Window>`**, type in the following **XAML**:

```
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition Height="Auto"/>
        <RowDefinition Height="*"/>
        <RowDefinition Height="Auto"/>
    </Grid.RowDefinitions>
    <StackPanel Grid.Row="0" Margin="25">
        <ComboBox Margin="5" Name="Options"
        HorizontalAlignment="Stretch"/>
        <TextBox Margin="5" PlaceholderText="Number"
        Name="Number" HorizontalAlignment="Stretch"/>
    </StackPanel>
    <CommandBar Grid.Row="3" VerticalAlignment="Bottom">
        <AppBarButton Icon="Accept" Label="Accept" Click="Accept_Click"/>
        <AppBarButton Icon="Cancel" Label="Clear" Click="Clear_Click"/>
    </CommandBar>
</Grid>
```

This **XAML** features a **`Grid`** with a **`StackPanel`** for the **`ComboBox`** and **`TextBox`** along with an **`AppBarButton`** to set or clear the **`BadgeNotification`** depending on which one was **Clicked**.

Step 8

Then, within **Solution Explorer** for the **Solution** select the arrow next to **MainWindow.xaml** then double-click on **MainWindow.xaml.cs** to see the **Code** for the **Main Window**.



Step 9

In the **Code** for **MainWindow.xaml.cs** there be a **Method** of **`myButton_Click(...)`** this should be **Removed** by removing the following:

```
private void myButton_Click(object sender, RoutedEventArgs e)
{
    myButton.Content = "Clicked";
}
```

Step 10

Once **`myButton_Click(...)`** has been removed, type in the following **Code** below the end of the **Constructor** of **`public MainWindow() { ... }`**:

```
private readonly Library _library = new();

private void Accept_Click(object sender, RoutedEventArgs e)
{
    _library.SetBadge(Options, Number);
}

private void Clear_Click(object sender, RoutedEventArgs e)
{
    _library.ClearBadge();
}
```

The **Methods** of **`Accept_Click`** and **`Clear_Click`** will call the **Methods** within *Library.cs* of **`SetBadge`** and **`ClearBadge`** respectively from an **Instance** of **`Library`** called **`_library`** created with **`new()`**.

Step 11

While still in the **Code** for **MainWindow.xaml.cs** within the **Constructor** of **`public MainWindow() { ... }`** and below the line of **`this.InitializeComponent();`** type in the following **Code**:

```
Options.ItemsSource = _library.Options;
Options.SelectedIndex = 0;
```

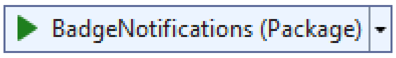
The **Constructor** of **`public MainWindow() { ... }`** should look like the following:

```
public MainWindow()
{
    this.InitializeComponent();
    Options.ItemsSource = _library.Options;
    Options.SelectedIndex = 0;
}
```

These set up the **Properties** for the **ComboBox** for **`ItemsSource`** to the list of **`Options`** from **`Library`** and for **`SelectedIndex`** to the first index which is **`0`** to select the first item.

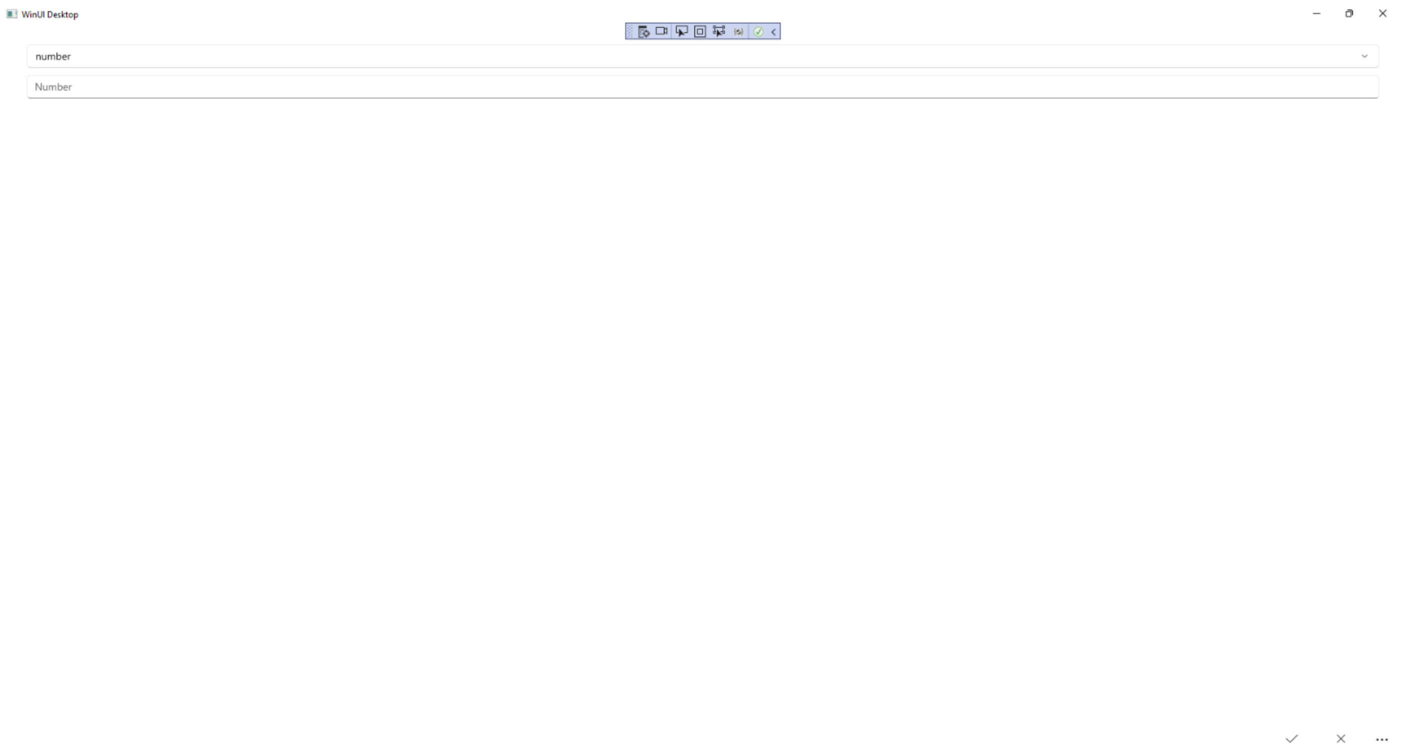
Step 12

That completes the **Windows App SDK** Application. In **Visual Studio 2022** from the **Toolbar** select **BadgeNotifications (Package)** to **Start** the Application.



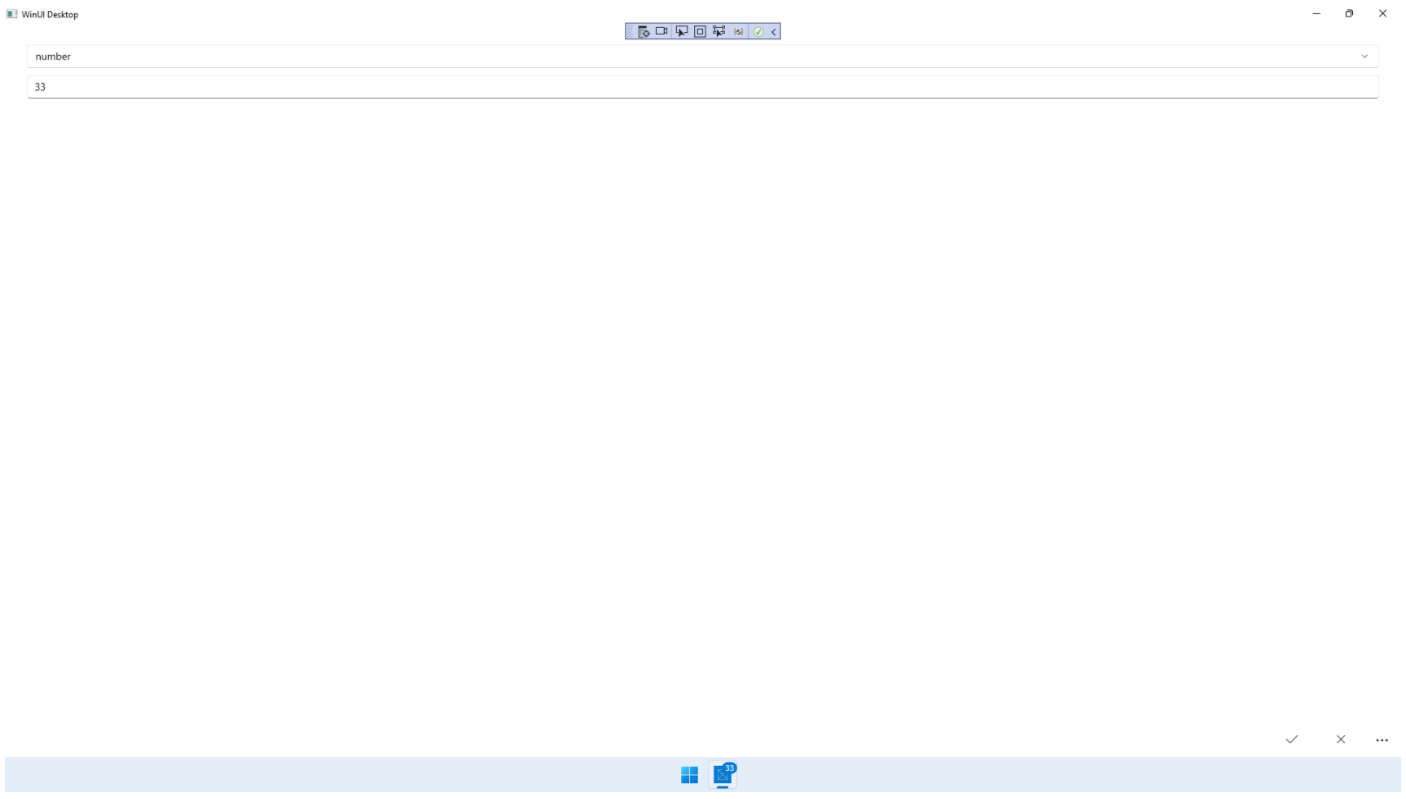
Step 13

Once running you should see the **`ComboBox`**, **`TextBox`** and **`CommandBar`** with *Accept* and *Clear* options.



Step 14

You can select a value from the **`ComboBox`** and then use *Accept* to see this on a **Badge** on the **Icon** for the Application in the **Taskbar** for example select n*umber* then enter a number or use *Clear* to reset the **Badge**.



Step 15

To **Exit** the **Windows App SDK** Application, select the **Close** button from the top right of the Application as that concludes this **Tutorial** for **Windows App SDK** from tutorialr.com!