



ActionMenus

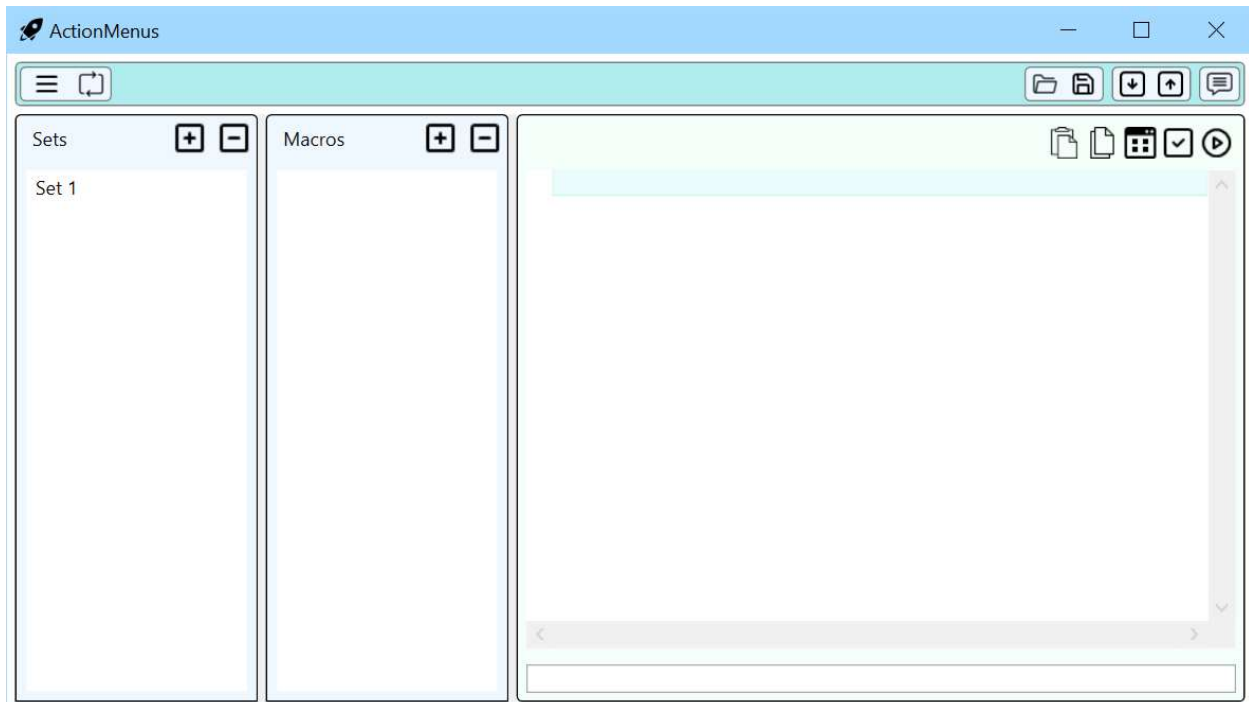
Introduction

This is a simple utility that has a lot of power and can be very dangerous. The basic concept is that this allows you to install a desktop extension that adds a menu item to the right click context menu that exposed a set of commands you can write in C#/.Net in the editor. These can do.. well.. anything, include destroy your computer.

So be careful.

The Bits








This is the entire UI component to ActionMenus:





The top bar is management functions. Sets are collections of related macros and you may add as many as you choose. To rename, double click the item and edit, then click out of the item or hit return. This applies to Macro names as well.

A Macro is a script which is edited in the script editor in the third column. Between each of the columns is a resize bar so you can resize the columns as you see fit.



1. Top bar

-  This is the menu preview. It shows you the current state of your design.
-  This forces a refresh to make sure the live menu (if installed) is current with your design
-  Open a saved setup
-  Save the current setup to a file
-  Install the extension
-  Uninstall the extension
-  Open the log file






2. Sets

-  Add new set
-  Remove currently selected set

3. Macros

-  Add new set
-  Remove currently selected set

4. Script Editor

-  Paste
-  Copy
-  Create a Visual Studio project with this script (for debugging)
-  Compile the script (errors appear in the text box at the bottom of the script editor)
-  Run the script.
 - Note, it will be run with no arguments.
 - In the next version, you will be able to set the debug arguments.

The Code Environment

Each new Macro starts with the same code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.IO;
using System.Threading.Tasks;
using System.Windows.Forms;

// Do not change namespace.
namespace ActionCode
{
    // Do not change class name.
    public class Program
    {
        // This gets called with list of selected files
        public static void Main(string[] args)
        {
            // add code here
        }
    }
}
```


The namespace and class name are fixed and cannot be changed. Changing them will cause your script to be not-runnable. As well, you must have a public, static method Main that takes one parameter, an array of strings and returns nothing.

When your script is called, a list of all the selected files and folders will be provided as the argument. If the user clicked on the desktop, then args will be empty.


Currently, the following assemblies are included for your script:

1. `System.dll`
2. `System.Data.dll`
3. `System.Xml.dll`
4. `System.Data.Linq.dll`
5. `System.Drawing.dll`
6. `System.Windows.Forms.dll`

In the next version, you'll be able to specify which assemblies you wish to include, including user created ones.

Since most scripts will be UI-less, errors are logged automatically to the ActionMenu log file. Clicking on  in the editor will open it in your default text editor.

However, since you have access to `System.Windows.Forms`, you can actually make your UI in code and display it. Each script is run in its own thread, and so remains alive until your script exits.

The script editor only allows very basic debugging at this time, so to make things easier – the  button allows you to take your current script and generate a complete Visual Studio 2017/2019 Community edition compatible solution. This lets you do full debugging in the VS environment. When you've got it working, copy the entire `Program.cs` file contents and replace your script's code with it.

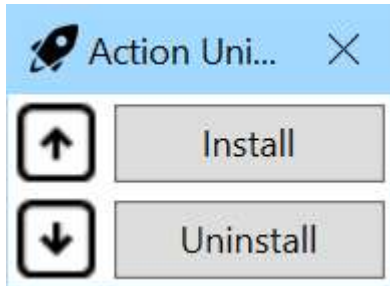
In a future version, this will be more integrated and you will be able to jump back and forth more easily.

Note, do not change the referenced assemblies. If you're getting an indication that the standard assemblies are missing, open the referenced assemblies folder in the Solution Explorer and click on any one of them. It should immediately correct the problem. This is a known bug and will be fixed as soon as possible.

The ActionUninstaller App

Because this application installs a COM object into the system, it's possible to get into a state where the editor cannot uninstall the COM object. To handle this case, the ActionUninstaller application can be used.

It's very simple:



I don't think much explanation is needed here.

Future Enhancements

1. A settings panel that will let you add assemblies, set the debug input and log files for each script.
2. A systray UI that lets you monitor the status of the extension and provides a way for scripts to communicate with the user.
3. Better script editor
4. Better integration with Visual Studio to use it as a satellite debugger and possibly editor.