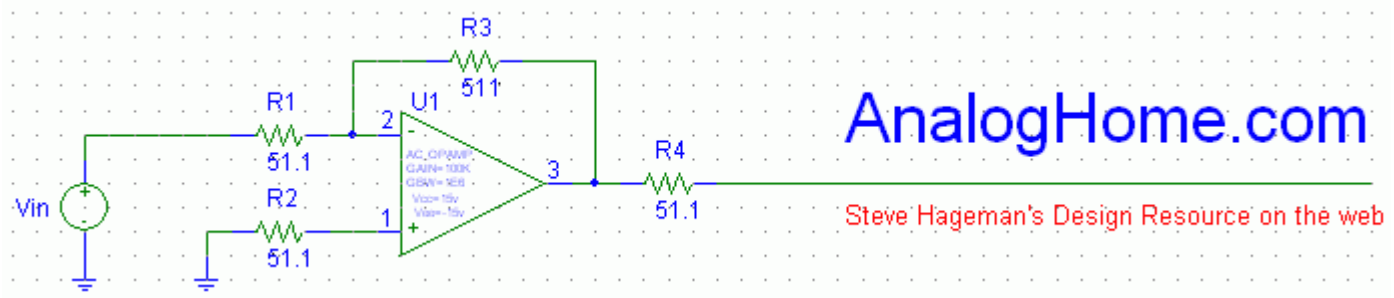

Sunday, March 27, 2022

Calling External Tools From Visual Studio 2022

(Random internet image grab of: Software Tools)

QT Designer is almost a substitute for the wonderful Drag and Drop Winform's GUI Interface that we have used for decades with Visual Basic and C# inside of Visual Studio.

The worst part is that they are not integrated well within any IDE, forcing you to go to a command prompt to run the Designer, then compile the UI code with the pyuic5.exe compiler. Linux folks are fine with running command-line tools all day long, but as a Windows user I don't want to have to remember command line prompts, so I wrote a script to automate the PyQT5 GUI building process.

The python script and methods for calling external programs from within Visual Studio apply to a wide range of possible uses. For instance, if one wanted to call and execute a GCC Make file, the same basic automation steps would be applicable.

Prerequisites:

PyQT5 Designer has been changing fast, so it is liable to change again by the time you read this (March 2022), so check online for the latest instructions.

To use PyQT5 Designer you have to have this package installed in your Python environment, either use a windows command prompt or if you use Anaconda, type this command at the Anaconda command prompt,

```
pip install PyQt5Designer
```

Automating with a Python Script:

While it is simple to launch a program from the Visual Studio Tools menu, automating a process where you need to select an item and then do some automation is not easy to do without a script of some sort. Python is the perfect language for these sorts of applications. It is relatively easy to get going on a windows PC and it is a well-known scripting language with millions of example scripts on the web to learn from.

The script that I wrote to automate the PyQT5uic compile process is available on GitHub [1]. Basically, that script, when run, opens a file Tkinter dialog box that asks you to select the PyQT5 Designer XML GUI file (the file with a .ui extension).

After a file is selected the script then assembles a command line like this,

```
pyuic5.exe "-x myGui.ui -o myGui.py"
```

Where myGui.ui is the GUI file that you make with the PyQt5 Designer program (it can be any name) and the myGui.py is the output file for the compiled .ui input file.

The '-x' command tells the compiler to make the resulting .py file 'executable', which means that it includes a: `if __name__ == __main_` block this option that allows you to execute the myGui.py file to see if it works immediately.

While sometimes this can be hard to get right using other languages, what with the need to make a string with the quotes in the proper spot and then call a shell to run it on the command line. In Python, there is this wonderful thing called "subprocess". It allows us to very simply just build a list of the commands to be output in sequence, then with one simple call to: "subprocess.run()" we get a perfect output every time.

```
# Run pyuic5.exe (It must be in the path somewhere)
cmd = []
cmd.append('pyuic5')
cmd.append('-x')

file_in = file_name + '.ui'
cmd.append(file_in)

cmd.append('-o')

file_out = file_name + '.py'
cmd.append(file_out)

result = subprocess.run(cmd)
```

subprocess.run() – Should be the heart of any Python Automation Script. With subprocess.run() a simple list is made up with the various command line parameters and then with a single call the external process is called with the parameters being perfectly ordered.

In the example below I downloaded and placed the automation script from GitHub [1] in the directory,

```
C:\Users\steve\AppData\Local\Programs\Python\Python39\Lib\site-
```

```
packages\QtDesigner\RunPyUic5WithPrompt.py
```

Your exact path will be slightly different. See below on how to find where your site-packages directory is if you don't know already.

Integrating PyQT5 Designer and Compiler Workflow

There is a myriad of videos on YouTube on how to step by step build a GUI using PyQT5, so I won't repeat that. What I want to do is to integrate the workflow into Visual Studio.

The process is,

1) Make a GUI page with Designer.exe, save it as an XML .ui file.
2) Compile the XML GUI file (.ui) to Python code file (.py)
3) Add the GUI Python code to your project.

Steps 1 and 2 above usually require the programmer to go to the command line to execute the various commands. That is annoying and I will show how to integrate those steps into the main Visual Studio Workflow here.

Visual Studio has a mechanism to add external tools to the Visual Studio IDE itself via the menu item: "Tools".

To add your Tools to this menu follow the steps below,

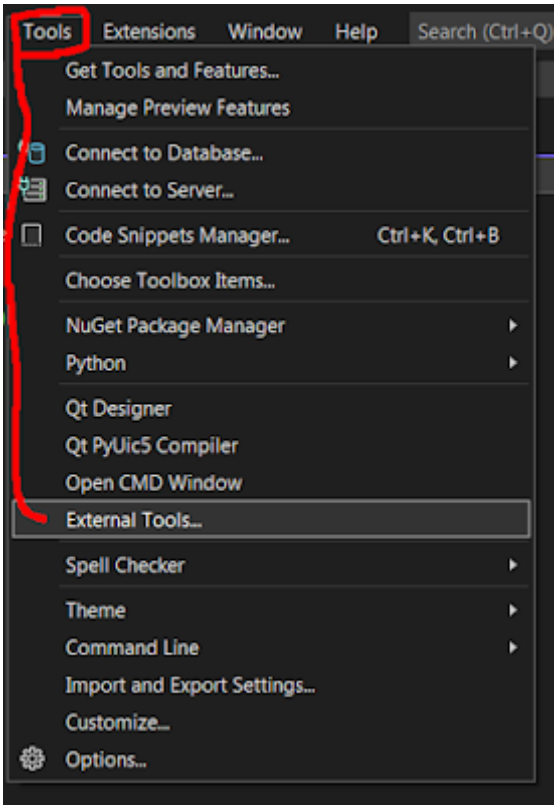

Figure 1 – Select Tools, then External Tools

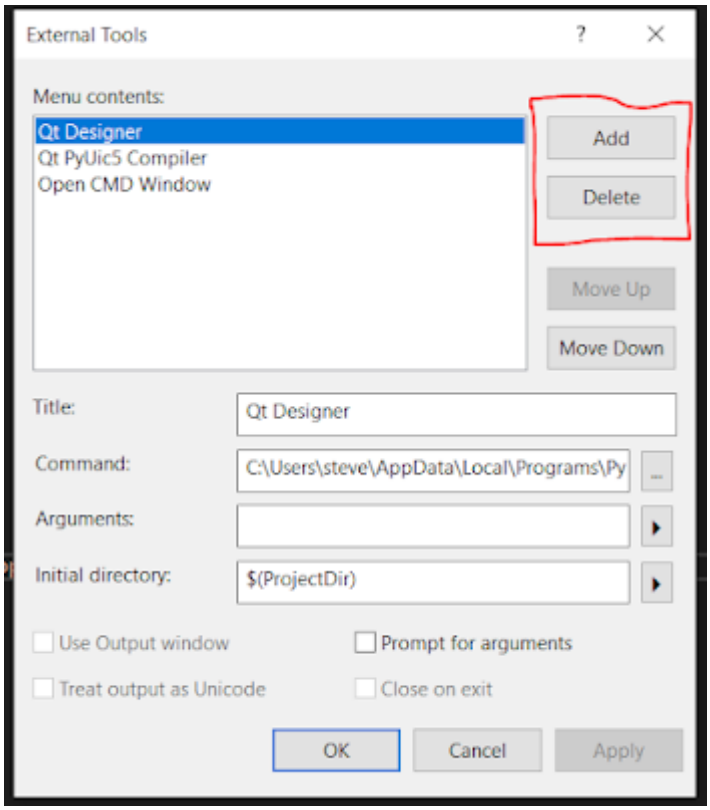

Figure 2 – In the External Tools dialog, you can add, delete or modify your add-in tools. Here I have three tools defined. I deleted the two default tools that Visual Studio ships with.

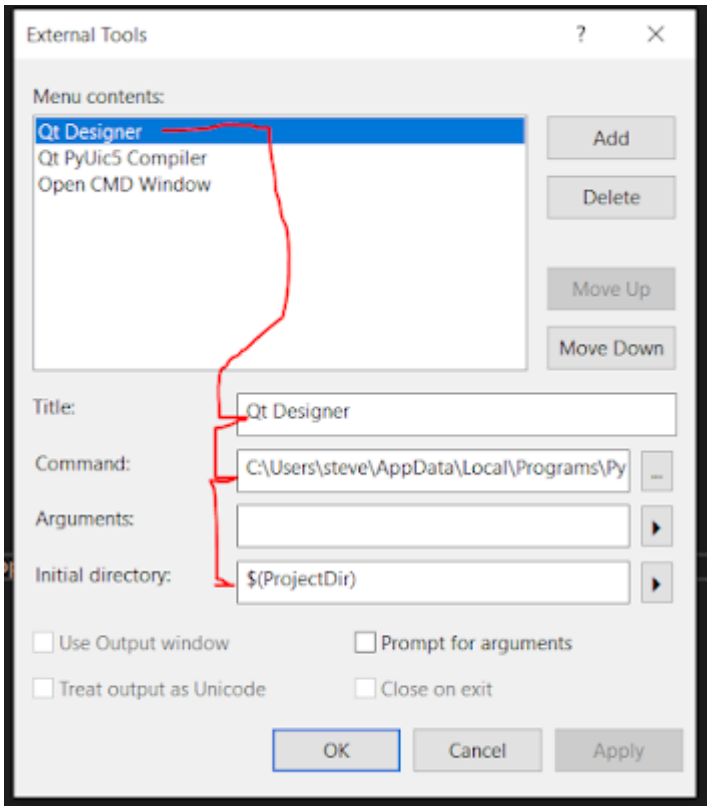

Figure 3 – To make the QT Designer entry, press “Add”, then add the following items,
1) I called the Title: “QT Designer” but you can use any title here you want.

2) In the Command, I added the path to the QT5 designer.exe program. In my case it is,

```
C:\Users\steve\AppData\Local\Programs\Python\Python39\Lib\site-packages\QtDesigner\designer.exe
```

Once you find where the designer.exe is on your system, then add that path instead.

Note: You can always find where your site-packages folder is by running this command at the command prompt,

```
python -c "import site; print(''.join(site.getsitepackages()))"
```

The initial directory is pointing to your current project as most people will want to keep the GUI code with the project. If not then select some other Initial Directory.

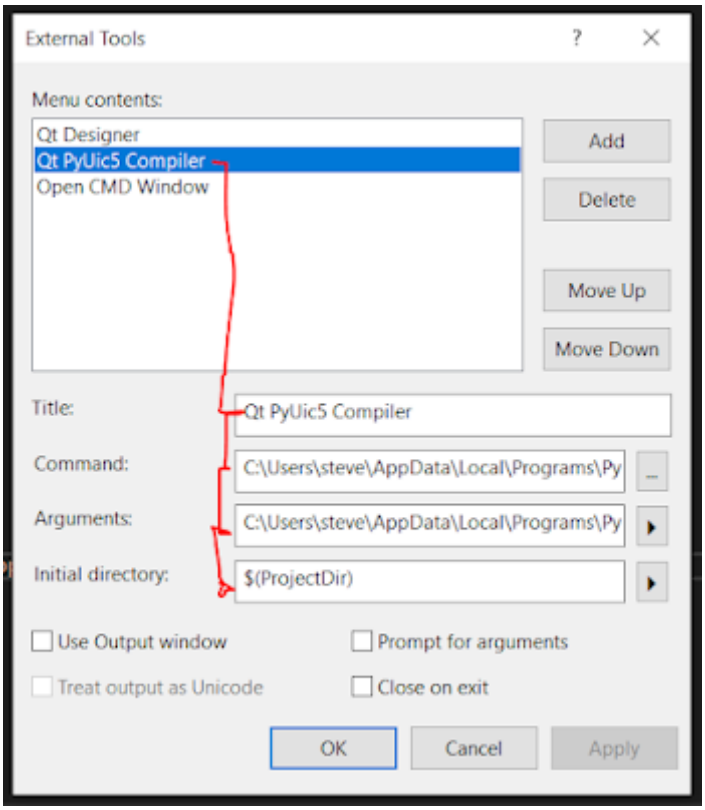

Figure 4 – Now "Add" another item for the PyQt5 UIC compiler – this is the program that converts the XML .ui file into Python code.
1) As above – Add any title that you like.
2) For the command, since this will be running a python script, we want to launch python itself, and in my system, this is the path to python (it will be slightly different for your system),

```
C:\Users\steve\AppData\Local\Programs\Python\Python39\python.exe
```

3) In the Arguments field add the following path to the automation script that we got from GitHub, Again this exact path will be slightly different on your system,

```
C:\Users\steve\AppData\Local\Programs\Python\Python39\Lib\site-packages\QtDesigner\RunPyUic5WithPrompt.py
```

4) The Initial Directory is again set to the project directory.

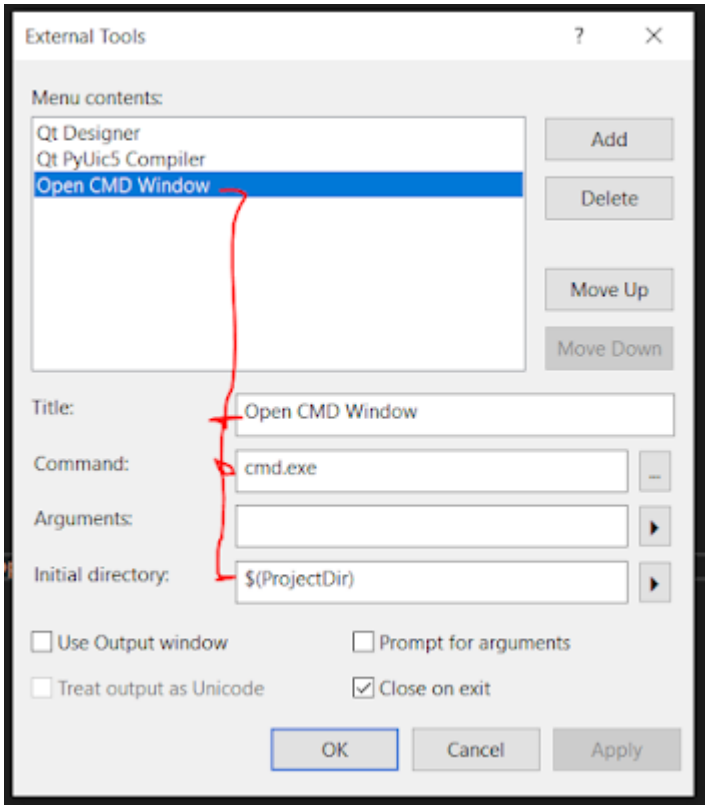

Figure 5 – As a bonus, since we are here, let's just add a tool that opens a command prompt in the current directory with the fields set as in the figure above.

Using The New Workflow:

Start a new Python Project in Visual Studio. To add a GUI element using the PyQT5 Designer, select: "Tools" → "Qt Designer" from the main menu. The PyQT5 Designer window should open.

Design your GUI as normal, then when you select "Save As" in the designer, it should be in your current working project directory. Save your new GUI element as a .ui file. Then exit the designer.

To compile your GUI .ui code file, again go to: "Tools" → "Qt PyUic5 Compiler" as shown in figure 6.

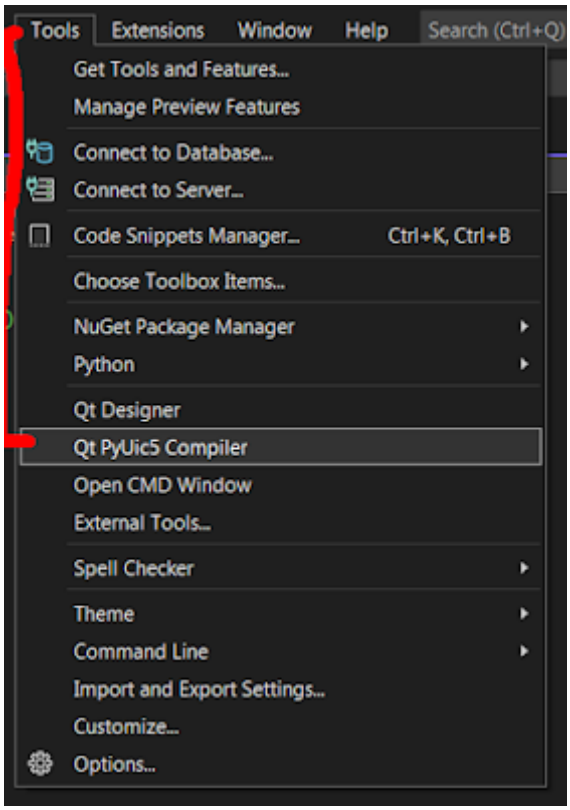

Figure 6 – To compile your .ui GUI files using the automation script, select: the "Tools" → Qt PyQTUic5 Compiler" item from the Visual Studio main menu.

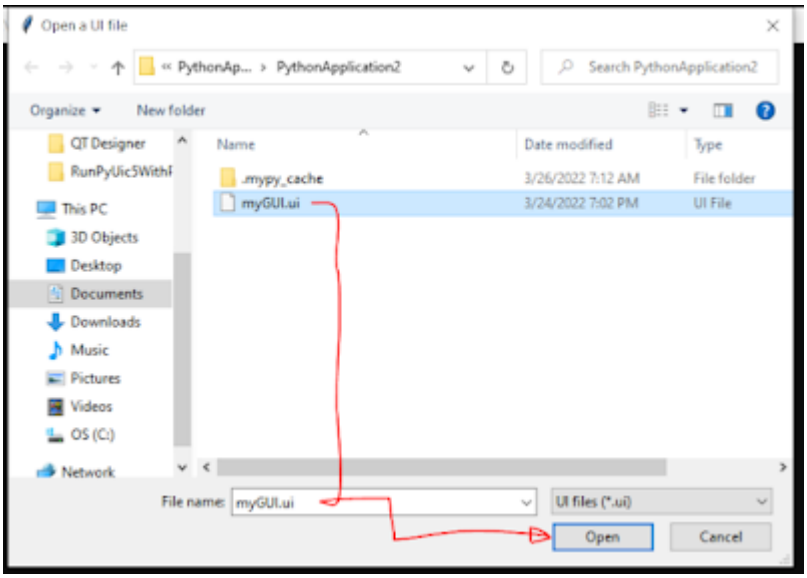

Figure 7 - The automation script that we added will pop up a dialog box asking you to select the .ui file that you wish to compile. Select the proper .ui file and then push "Open".

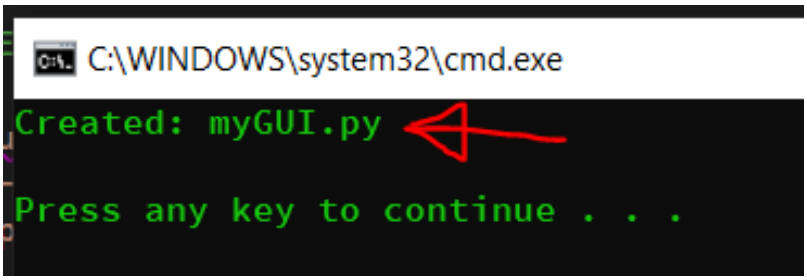

Figure 8 – The PyQtUic5.exe program will then be called on the selected .ui file and compile to a .py file. When the script completes it will say that it is finished. If some error happens during compilation, that will also show up in the Python command window.

Warping it up:

You can easily extend Visual Studio by writing automation scripts and then including the workflow commands into the Visual Studio External Tools to make any process much more user-friendly.

With the automation here, you can then start a Python project, design a PyQT5 GUI, and compile it all without having to leave the Visual Studio IDE.

Almost all IDE's have the hooks to add calls to external tools, so the concept presented here can be implemented in a wide variety of IDE's and programming environments.

Bonus Information:

While PyQT5 is the tool to use for large GUI's, it still leaves a lot to be desired and is not as easy to use as the Visual Studio Winform's designer.

In those instances where just a file or general dialog box are needed then it may be better to just use the base functionality in Tkinter itself (Also see the python automation script that I wrote) [2].

If a slightly more complex GUI is needed, then consider PySimpleGUI [3]. This is a simple row and column-based way to make a GUI. It is made simple to use by the hundred or more demo programs provided on the PySimpleGUI GitHub page [4]. Just find something close to what you need and clone it as a starting point – that IS simpler than the Visual Studio WinForm's designer.

References:

[1] The Python automation script presented here can be found on GitHub at,
https://github.com/Hagtronics/PyQt5-Designer-Automation-Script

[2] Tkinter documentation
https://docs.python.org/3/library/tkinter.html

[3] PySimpleGUI home on GitHub,
https://github.com/PySimpleGUI/

[4] PySimpleGUI Demo programs,
https://github.com/PySimpleGUI/PySimpleGUI/tree/master/DemoPrograms