AU 2023
THE DESIGN & MAKE CONFERENCE

MFG601910

Bridging the Gap Between iLogic Automation and Inventor Add-Ins

Curtis Waguespack
Team D3

Learning Objectives

- Create simple add-ins using your existing iLogic rules.
- Demystify the “how to get started” process of Inventor automation add-ins.
- Learn to create a richer user interface for your automation tools.
- Better understand the Inventor add-in tools, templates, and techniques.

Description

You’ve seen how iLogic automation can tame tedious and error-prone tasks and make you and your fellow Inventor users more efficient.

But have you also run up against the limitations present within the automation tools offered with iLogic? Have you struggled to take things to the next level and create Inventor add-ins? Are you concerned about having to start over?

You’re not alone. Making the transition from iLogic is a difficult transition for many busy professionals who want to take their automation further but find themselves sticking with what is known, what is familiar, and what is already in place.

Let’s have a look at what you need to get started on a path forward that allows you to leverage the familiarity of your existing automation while learning the landscape of the Inventor add-in.

Author

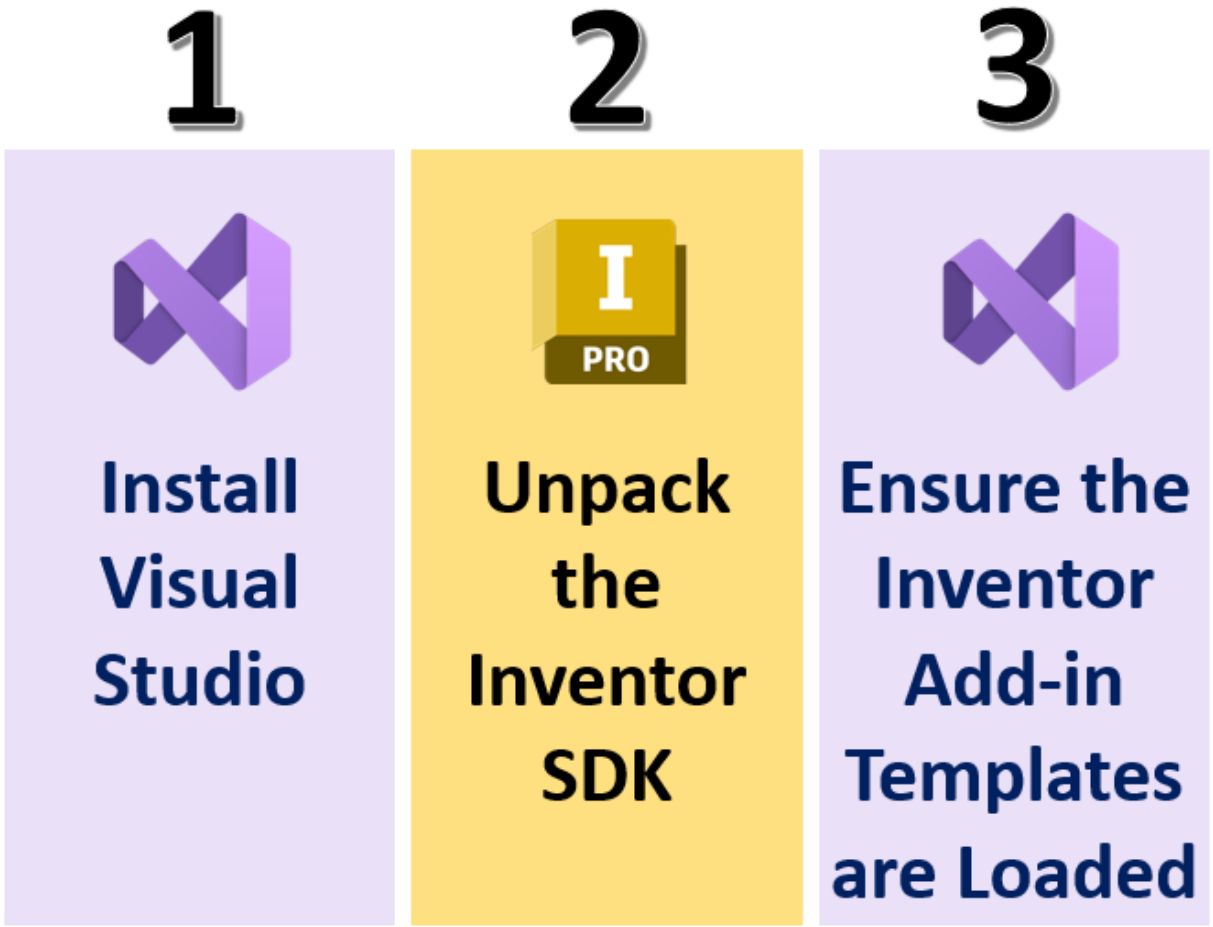
Curtis has used Autodesk Inventor and AutoCAD for over two decades to design a wide range of manufactured products. In addition to his design experience, Curtis has authored and co-authored multiple editions of the Mastering Autodesk Inventor book and has taught Inventor to professionals in the classroom. Currently, Curtis works as an Automation Solutions Consultant, where his focus is on automation within Autodesk Inventor and helping others in the Inventor community understand and utilize iLogic and Inventor API automation. Curtis creates Inventor automation tools that help companies become more efficient with their design efforts, and he teaches iLogic classes as well.

AU 2023
THE DESIGN & MAKE CONFERENCE

Getting Started Steps

AU 2023
THE DESIGN & MAKE CONFERENCE

These are the three general steps to getting started Visual Studio and the Inventor add-ins templates. This document will take you through these steps.



AU 2023
THE DESIGN & MAKE CONFERENCE

Installing Visual Studio

Go to the link below and download and install Microsoft Visual Studio.
https://learn.microsoft.com/en-us/visualstudio/ide/?view=vs-2022

Determine the version to use

You will need to determine the version of Visual Studio to download and use.

- If your intention is to use it for educational purposes, you can use the Community version.
- If you are intending to use it for production or profit you will need to use the Professional version.

Visual Studio Professional

- Full version
- Purchased from Microsoft
- Full Use Licensing

Visual Studio Community

- Full Version
- Free from Microsoft
- Limited Use Licensing

~~Visual Studio Express~~

- ~~Limited version~~
- ~~Free from Microsoft~~
- ~~Not suitable for Inventor add-ins~~

AU 2023
THE DESIGN & MAKE CONFERENCE

Unpack the Inventor SDK files

The Inventor SDK (Software Developer Kit) installs with Inventor automatically. However, you will need to "unpack" the contents of the **developertools.msi** file to use the SDK contents.

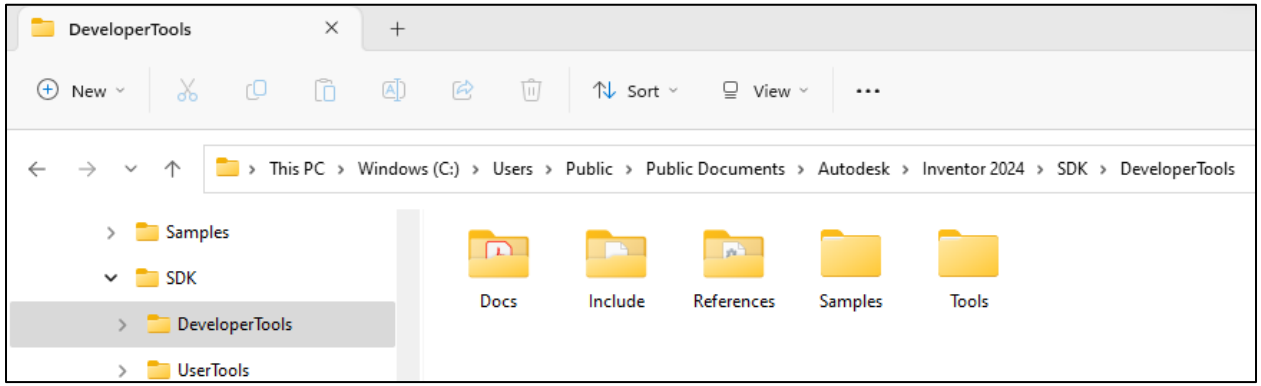
- In Windows Explorer go to:
  - C:\Users\Public\Documents\Autodesk\Inventor 2024\SDK
- Double-click the **developertools.msi** file
- Click the **Next** button
- Check the **Accept Terms** box
- Click the **Next** button
- Choose the path
- Click the **Next** button
- Click the **Install** button
- Click the **Finish** button

Note that you need to have Visual Studio Installed before you unpack the developertools.msi file.

Examine the DevelopersTools folder

You will now have a DevelopersTools folder containing the subfolders shown.

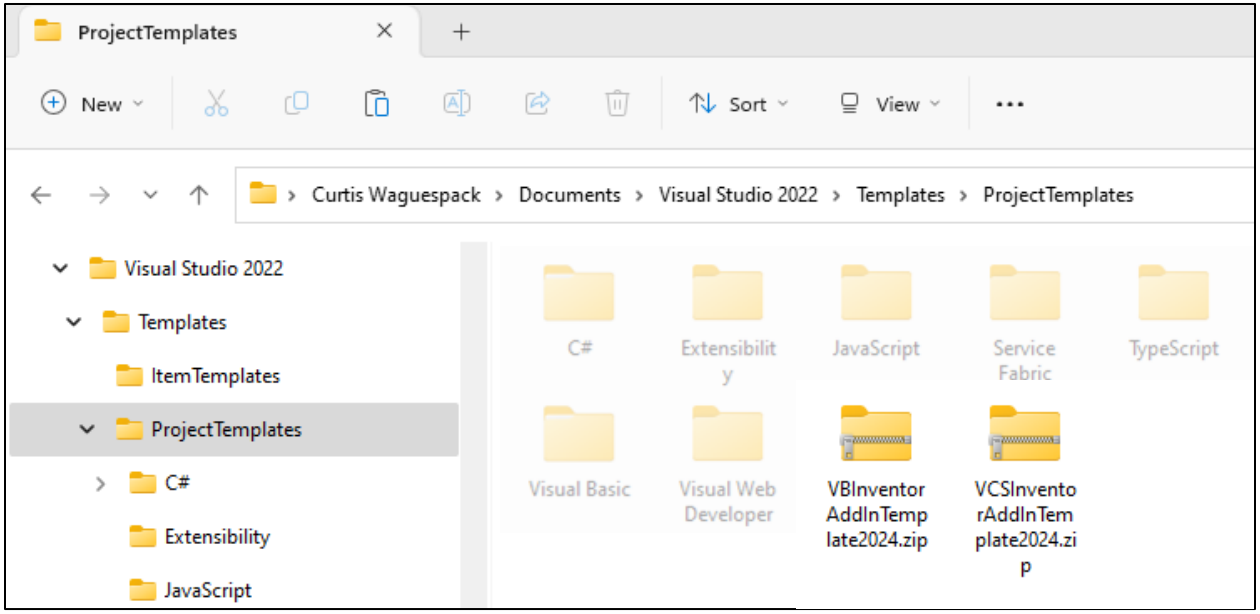
C:\Users\Public\Documents\Autodesk\Inventor 2024\SDK\DevelopersTools



Examine the Visual Studio Templates

In addition to the DevelopersTools folder the developertools.msi will “unpack” some Visual Studio template files. Typically, the location of these templates something like this:

C:\Users\CurtisWaguespack\Documents\Visual Studio 2022\Templates\ ProjectTemplates



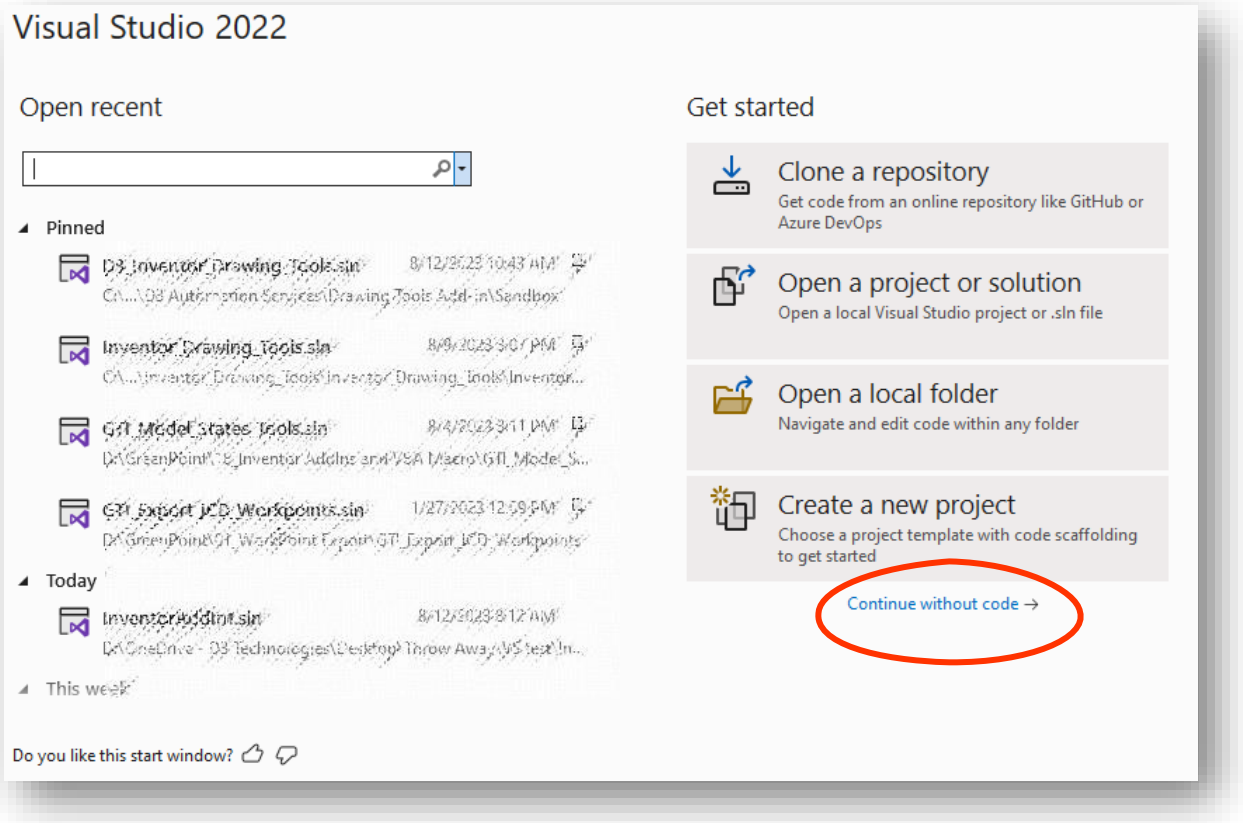
Ensure the Inventor Add-In Templates show up in Visual Studio

In the following section you will ensure the Inventor Add-in templates are loaded in Visual Studio

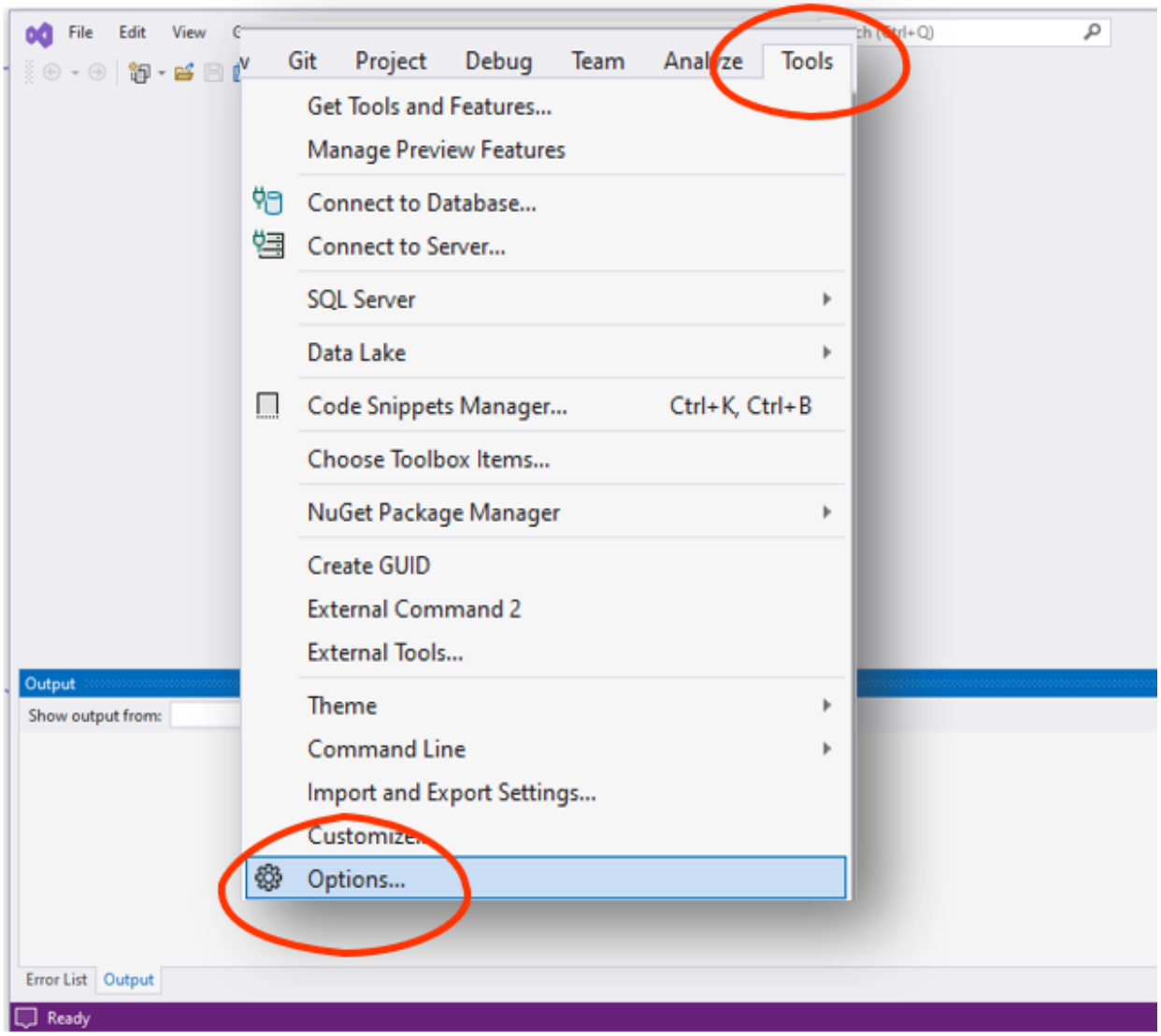
Ensure the Inventor Add-In Templates show up in Visual Studio

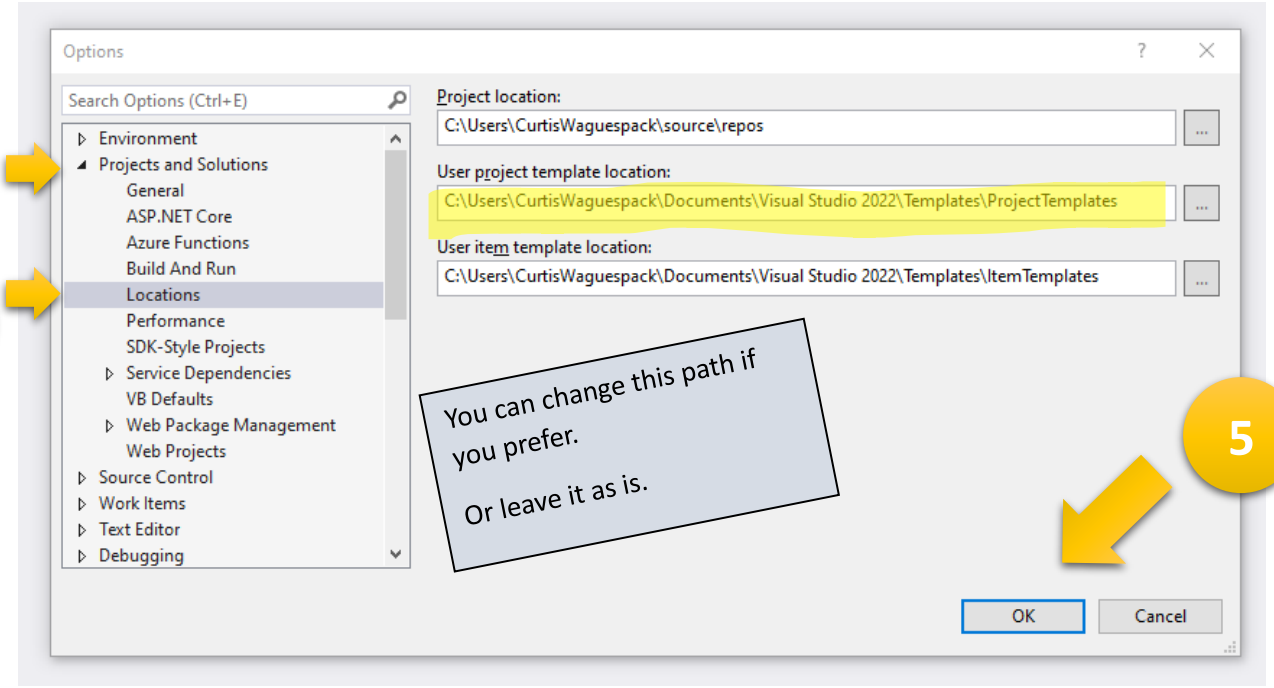
Next you will open Visual Studio and check the template location.

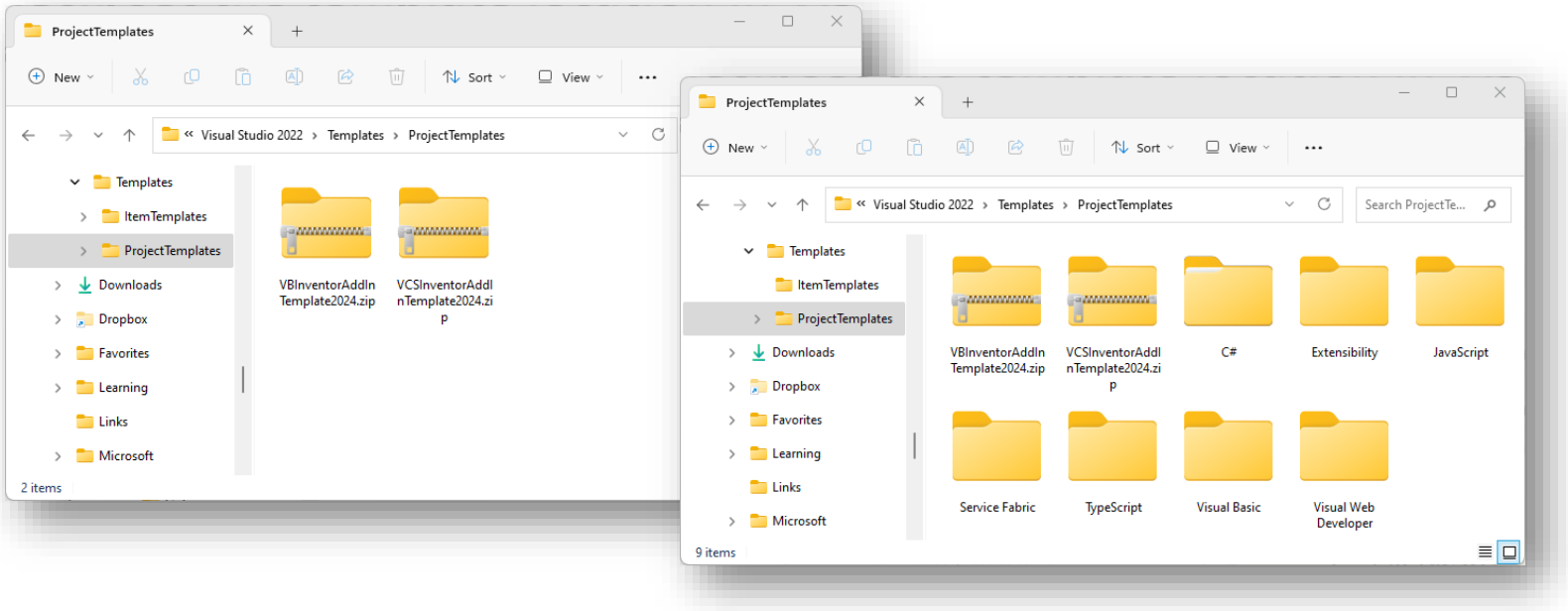
1. Open Visual Studio > click **Continue without code**
2. In Visual Studio click the **Tools** tab > select **Options** from the menu
3. Select **Projects and Solutions** in the left pane
4. Click **Locations**
5. Click the **OK** button to create the standard folders



2


File
Edit
View
Git
Project
Debug
Team
Analyze
Tools
Get Tools and Features...
Manage Preview Features
Connect to Database...
Connect to Server...
SQL Server
Data Lake
Code Snippets Manager... Ctrl+K, Ctrl+B
Choose Toolbox Items...
NuGet Package Manager
Create GUID
External Command 2
External Tools...
Theme
Command Line
Import and Export Settings...
Customize...
Options...
Output
Show output from:
Error List
Output
Ready

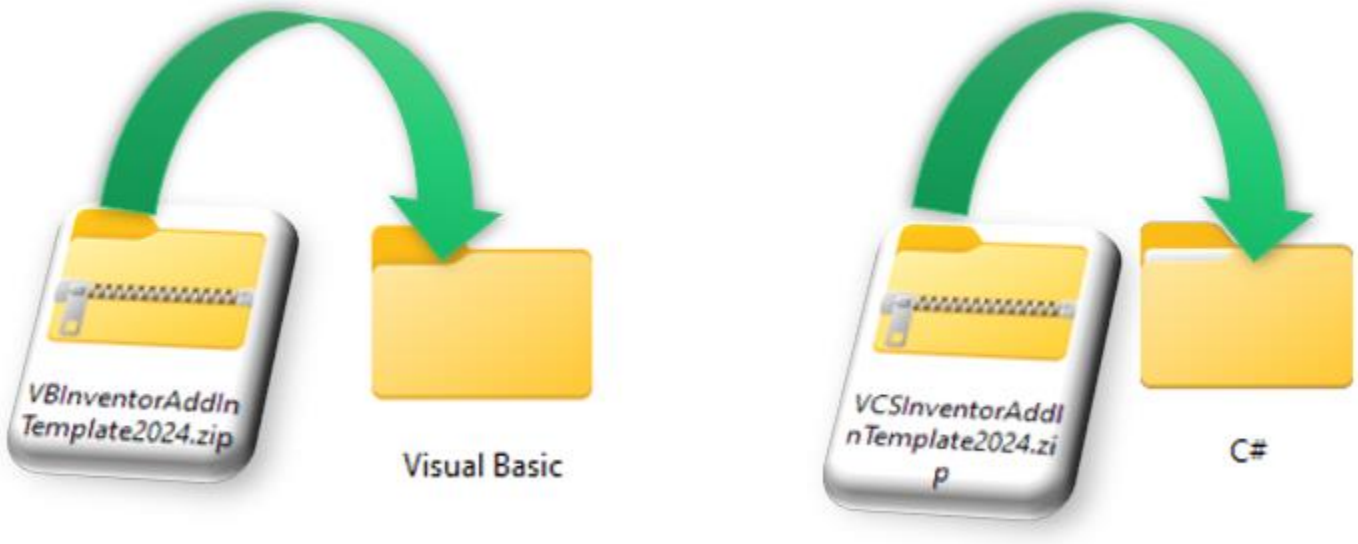

Options
Search Options (Ctrl+E)
Environment
Projects and Solutions
General
ASP.NET Core
Azure Functions
Build And Run
Locations
Performance
SDK-Style Projects
Service Dependencies
VB Defaults
Web Package Management
Web Projects
Source Control
Work Items
Text Editor
Debugging
Project location:
C:\Users\CurtisWaguespack\source\repos
User project template location:
C:\Users\CurtisWaguespack\Documents\Visual Studio 2022\Templates\ProjectTemplates
User item template location:
C:\Users\CurtisWaguespack\Documents\Visual Studio 2022\Templates\ItemTemplates
OK
Cancel
3
4
5
You can change this path if you prefer.
Or leave it as is.
Even if you do not change the path, it is a good idea to click the OK button, and some standard folders will be created.


ProjectTemplates
New
Sort
View
Visual Studio 2022 › Templates › ProjectTemplates
Templates
ItemTemplates
ProjectTemplates
Downloads
Dropbox
Favorites
Learning
Links
Microsoft
2 items
VBInventorAddInTemplate2024.zip
VCSInventorAddInTemplate2024.zip
C#
Extensibility
JavaScript
Service Fabric
TypeScript
Visual Basic
Visual Web Developer
9 items

Move the *.zip files into the standard folders

You can browser to he template folder and move the *.zip file to their respective folders. This isn't required, but is best practice.

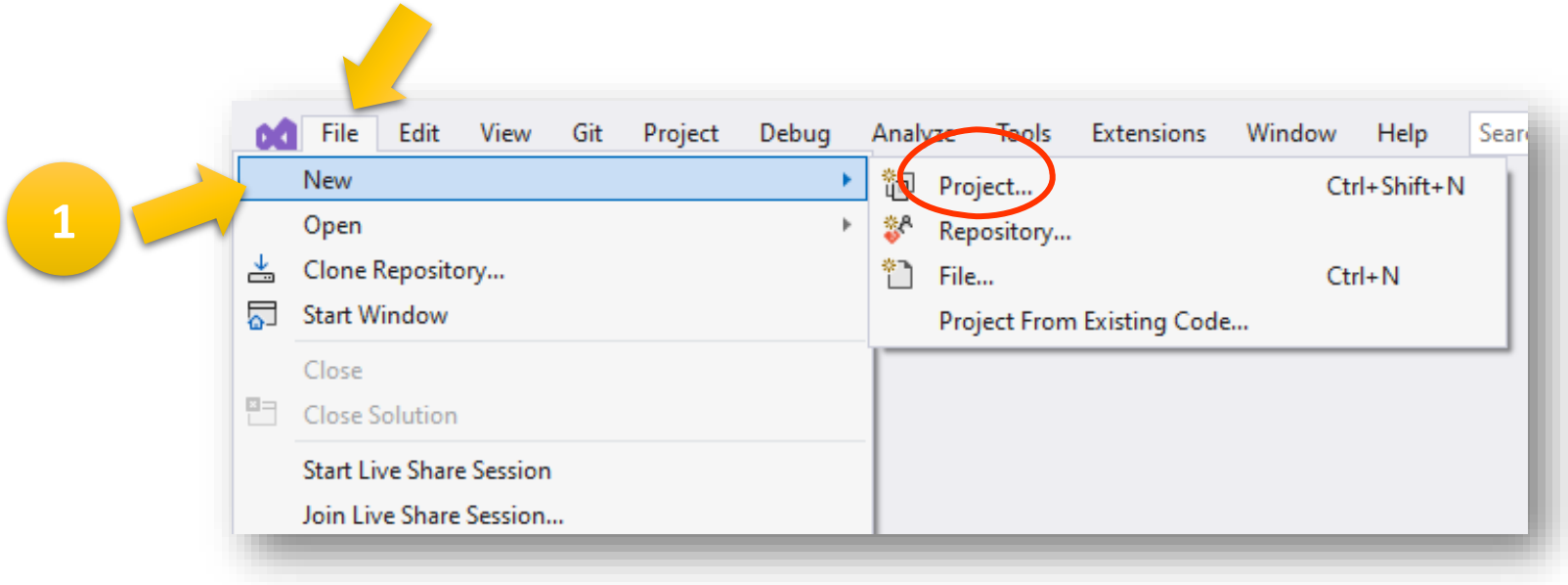
- Move the **VBInventorAddinTemplate** zip file to the **Visual Basic** folder
  - You don't need to unzip it
- Move the **VCSInventorAddinTemplate** zip file to the **C#** folder
  - You don't need to unzip it

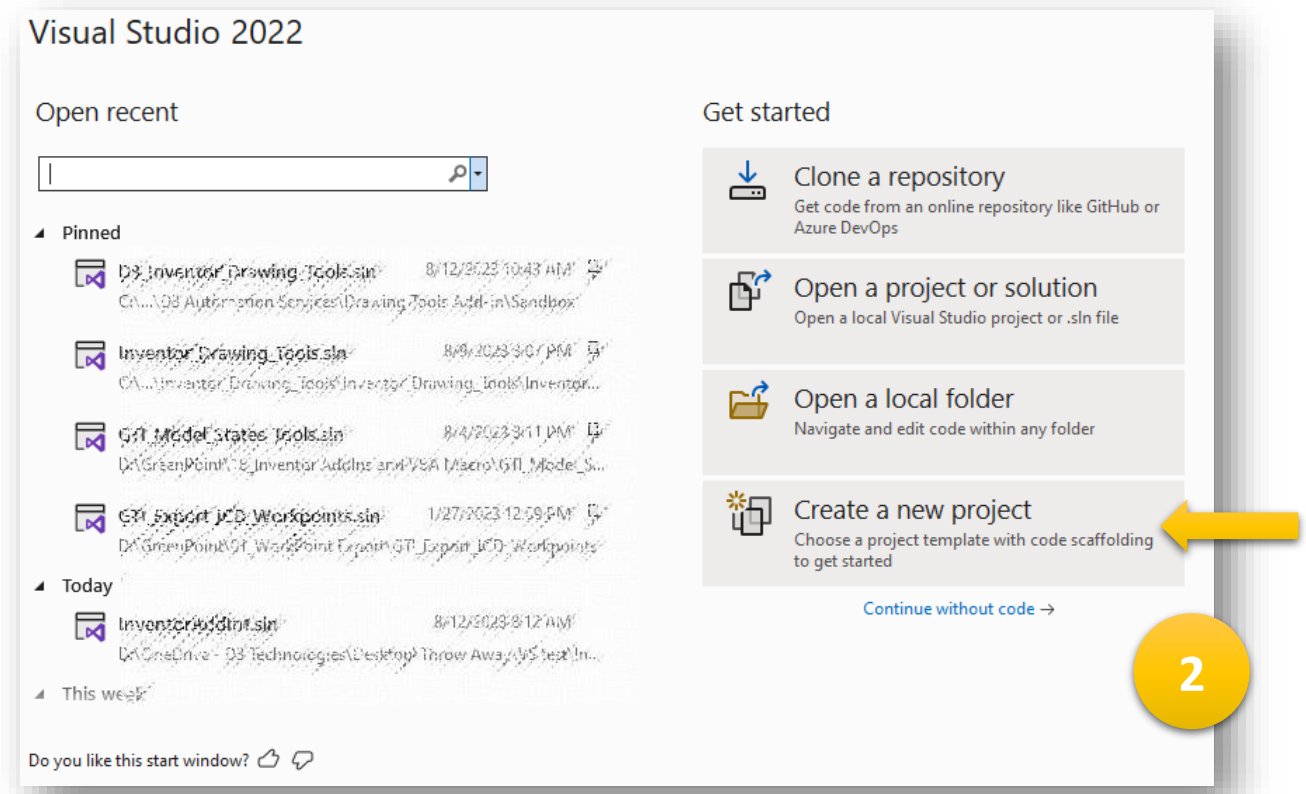


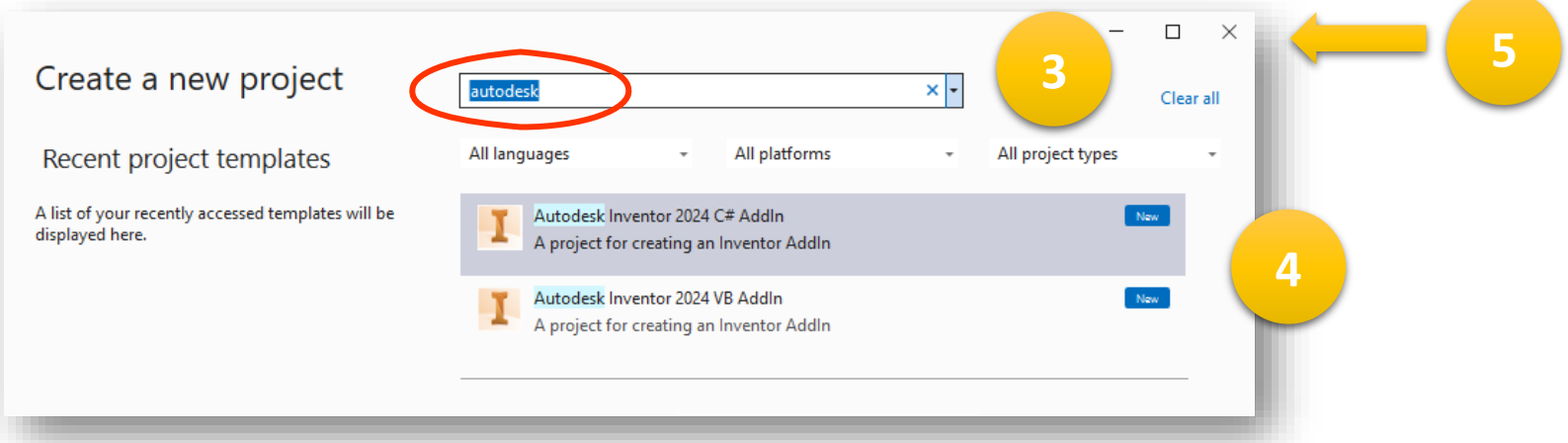
Ensure the templates show up

Next you will go through the steps to create a new Visual Studio project in order to check that the *.zip file templates show up in the Create New Project dialog.

1. Still in Visual Studio click the **File** tab > select **New** from the menu.
2. Select **Create New Project** in the resulting dialog.
3. Type **Autodesk** or **Inventor** int the search bar
4. Note the templates in the list
5. Click the X in the top right corner to exit out of the Create New Project dialog.
    - This step was just to ensure the add-ins loaded, you will create a new project using a custom add-in in the steps to come.



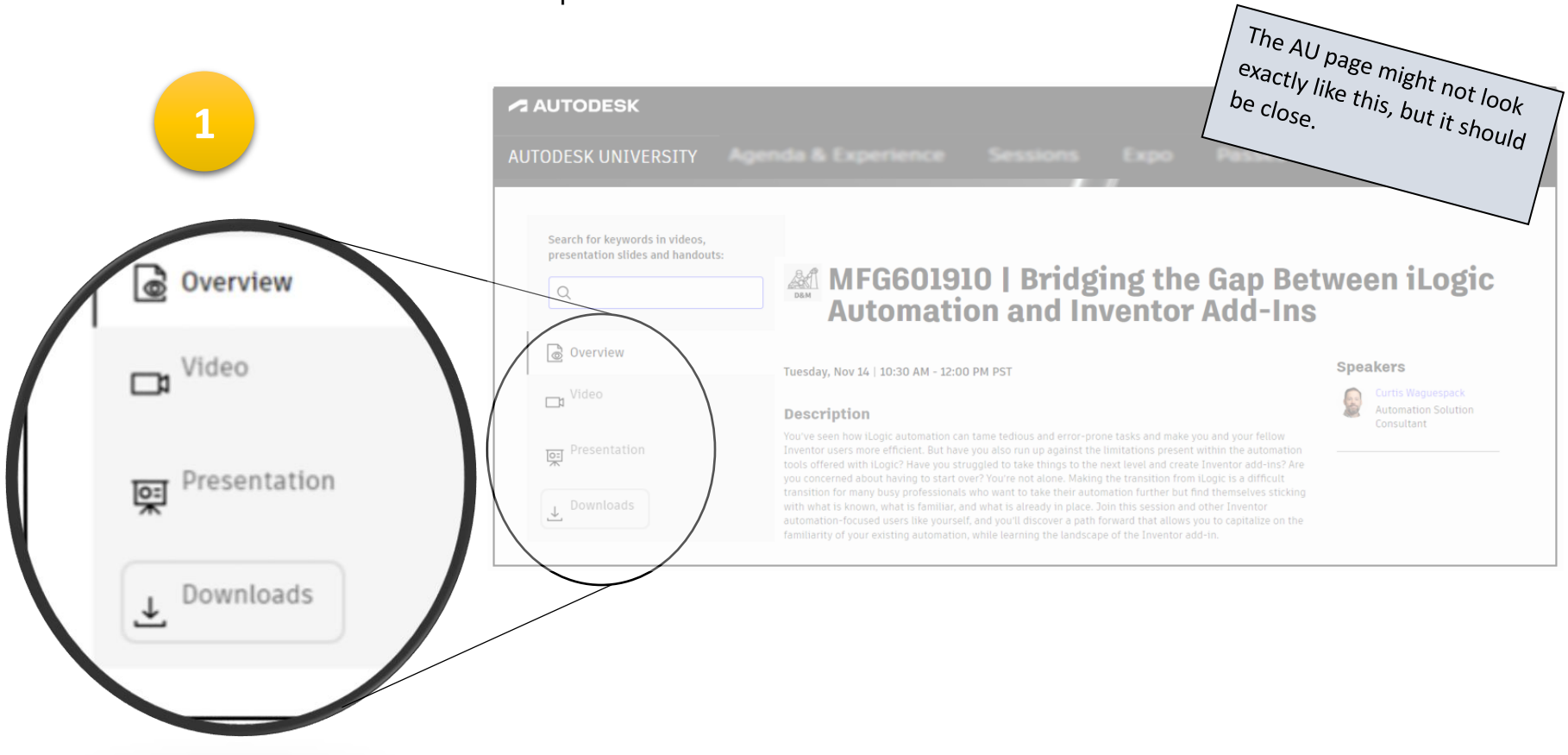

Visual Studio 2022
Open recent
Pinned
D3_Inventor_Drawing_Tools.sln
Inventor_Drawing_Tools.sln
GT_Model_States_Tools.sln
GT_Export_iCD_Workpoints.sln
Today
InventorAddIn.sln
This week
Do you like this start window?
Get started
Clone a repository
Get code from an online repository like GitHub or Azure DevOps
Open a project or solution
Open a local Visual Studio project or .sln file
Open a local folder
Navigate and edit code within any folder
Create a new project
Choose a project template with code scaffolding to get started
Continue without code →
2


Create a new project
autodesk
3
Clear all
5
Recent project templates
A list of your recently accessed templates will be displayed here.
All languages
All platforms
All project types
Autodesk Inventor 2024 C# AddIn
A project for creating an Inventor AddIn
New
Autodesk Inventor 2024 VB AddIn
A project for creating an Inventor AddIn
New
4

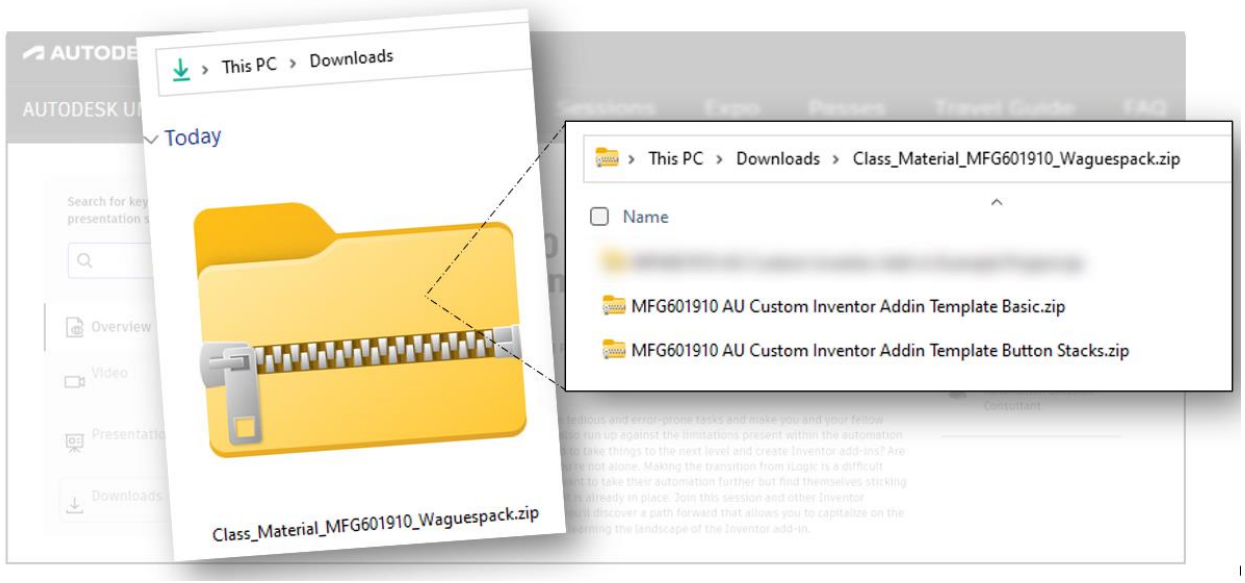
Downloading the custom add-in template

This AU class provides some custom add-in templates to get you going with Inventor add-ins. You will download these from the AU web site for this class (*MFG601910 - Bridging the Gap Between iLogic Automation and Inventor Add-Ins*).

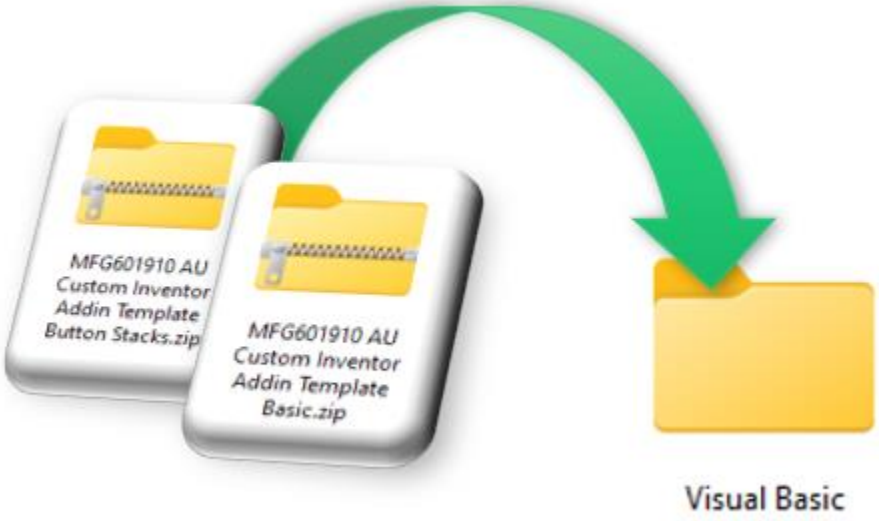
1. Find this AU class online, and download the class materials.
2. Find the templates within the class materials.
    - The exact file count and names might vary from the images in this guide by the time the download materials are released.
3. Move the template zip files to the Visual Basic folder.
    - C:\Users\CurtisWaguespack\Documents\Visual Studio 2022\Templates\ ProjectTemplates\Visual Basic
    - You don't need to unzip them.



2


This PC › Downloads
Today
Class_Material_MFG601910_Waguespack.zip
This PC › Downloads › Class_Material_MFG601910_Waguespack.zip
Name
MFG601910 AU Custom Inventor Addin Template Basic.zip
MFG601910 AU Custom Inventor Addin Template Button Stacks.zip

3

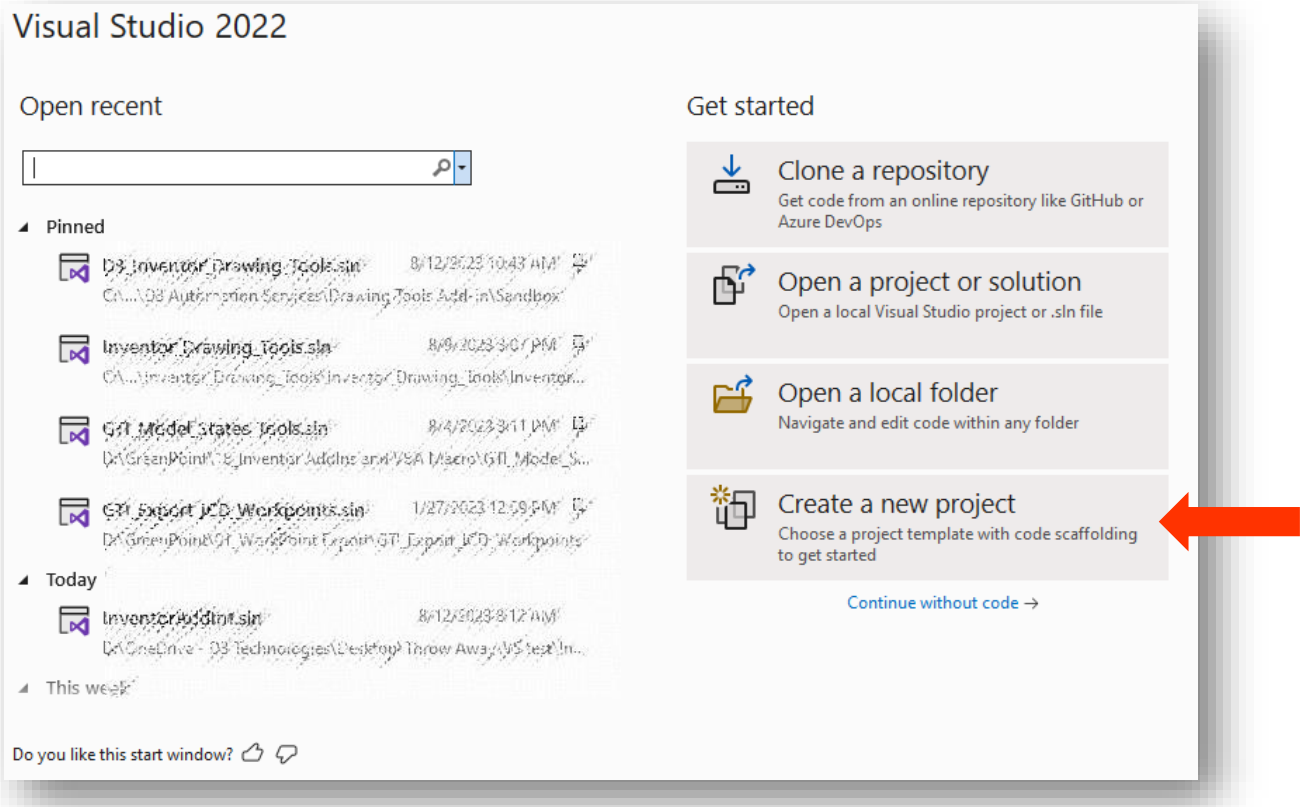

MFG601910 AU Custom Inventor Addin Template Button Stacks.zip
MFG601910 AU Custom Inventor Addin Template Basic.zip
Visual Basic

Creating an Add-in from an improved template

Getting Started

Creating an Add-in

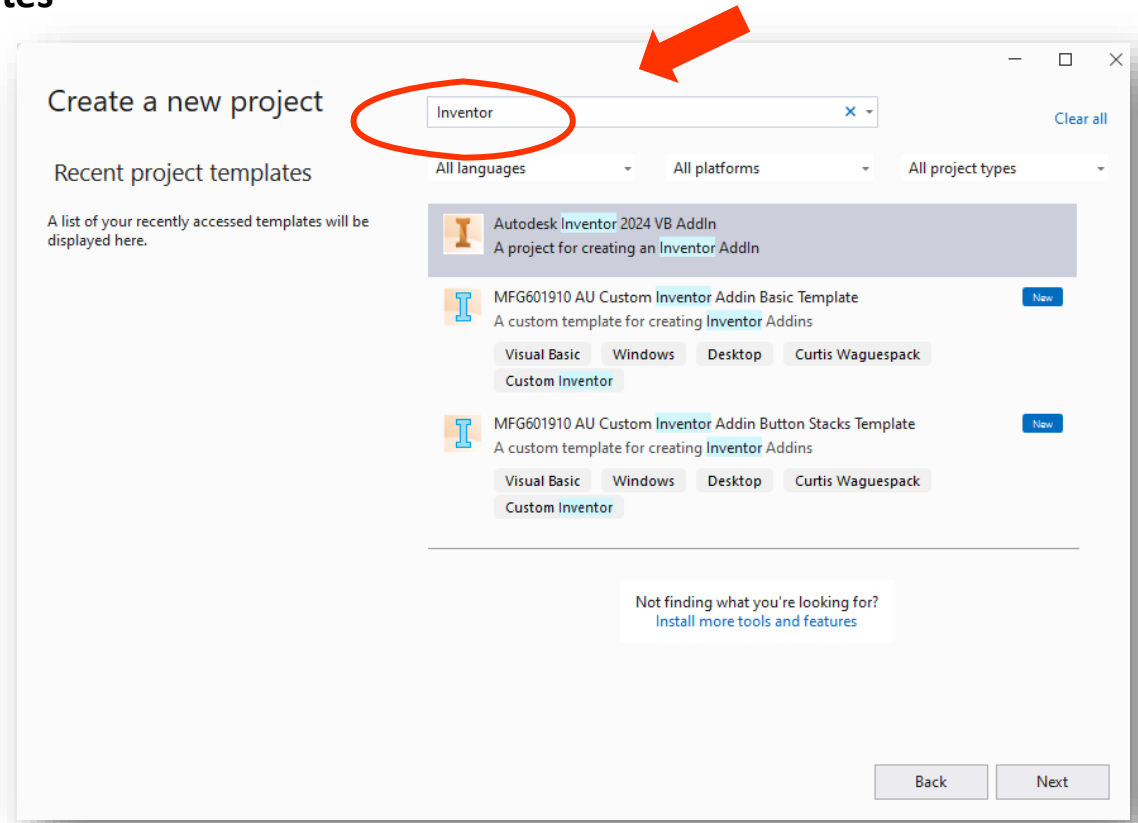
Click Create a New Project



Creating an Add-in

Find the Inventor Add-in Templates

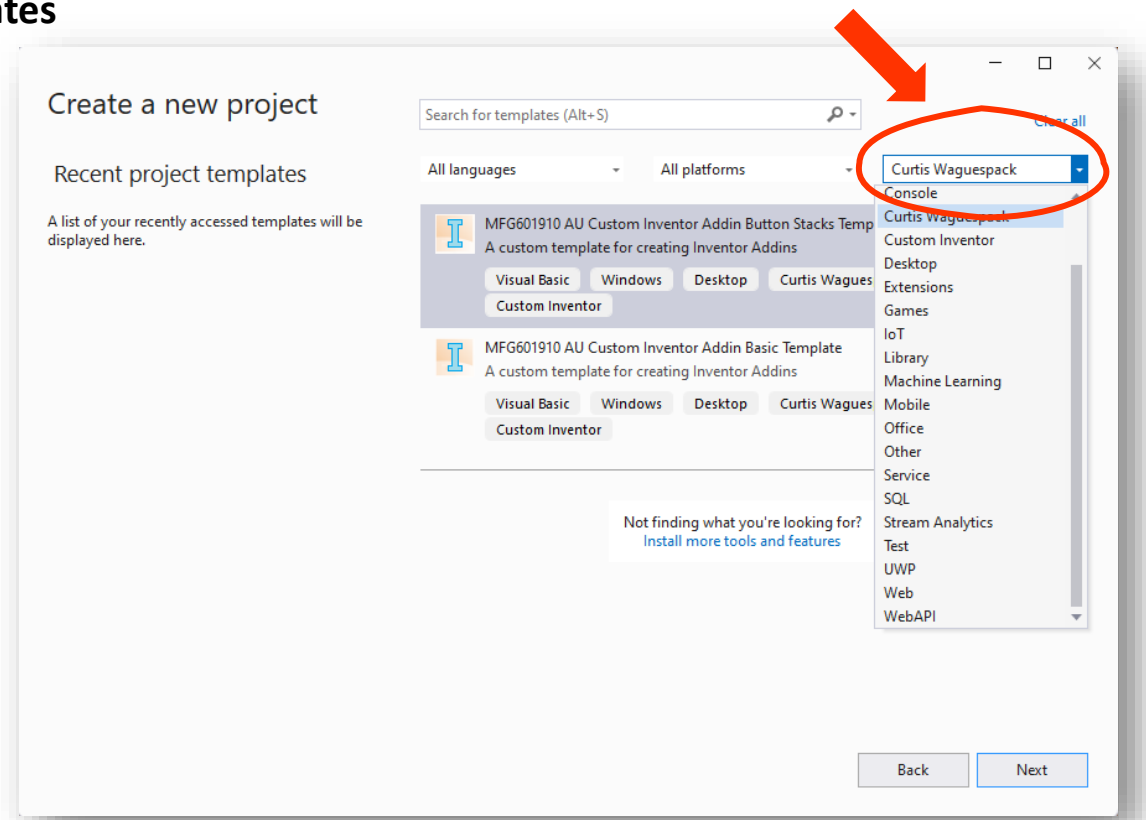
- Type “Inventor” in the search bar to filter for just the templates you want to use
- You should see the templates show up, similar to this image.



Creating an Add-in

Find the Inventor Add-in Templates

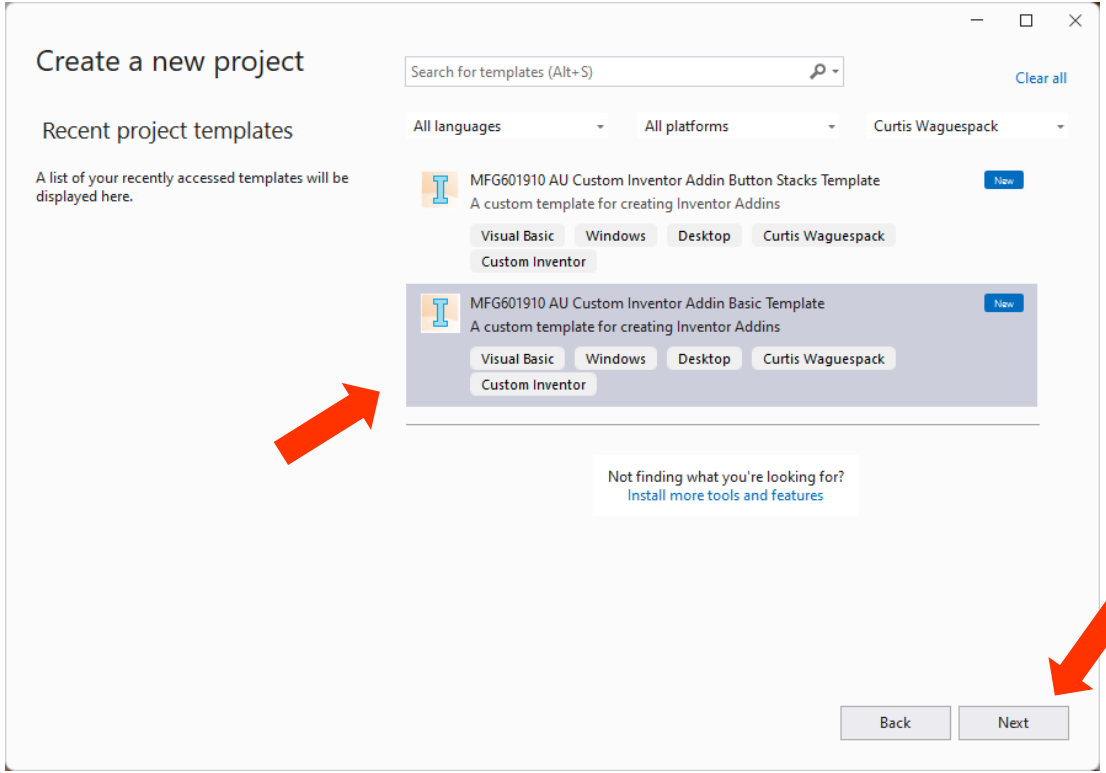
- Or you can use the filter drop-down to filter for just the add-ins provided with the class materials by selecting my name from the list



Creating an Add-in

Find the Inventor Add-in Templates

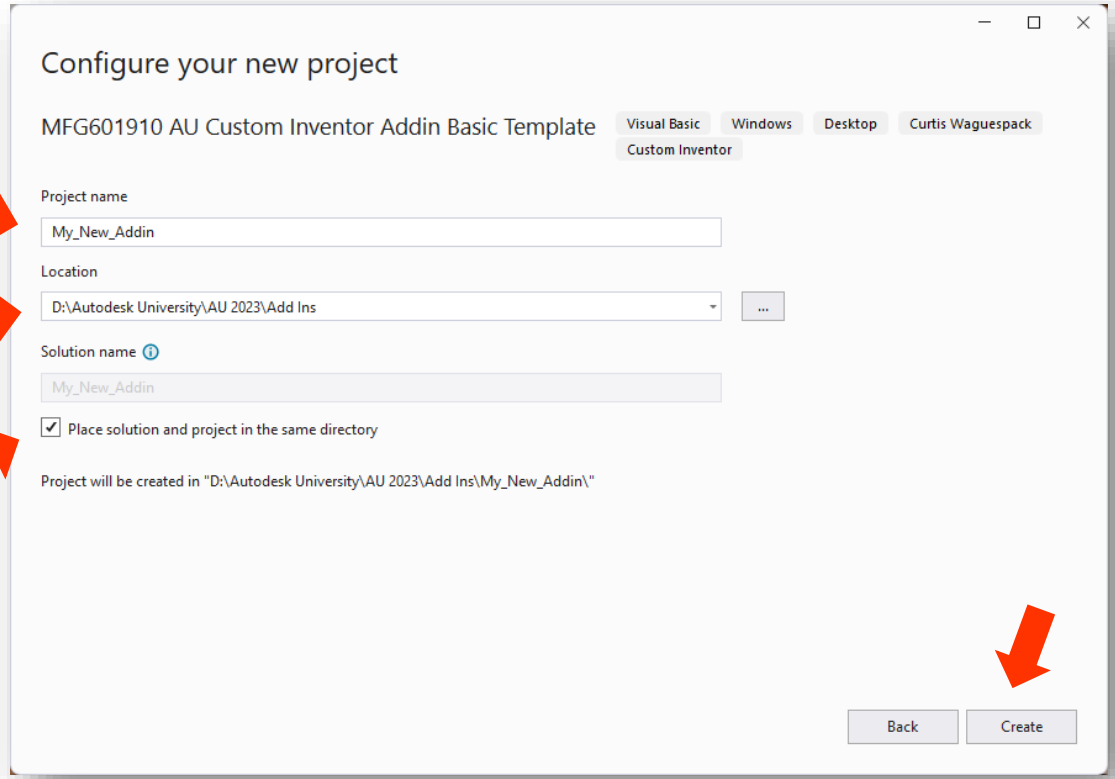
- Select the template called: **MFG601910 AU Custom Inventor Addin Template Basic**
- Click the **Next** button



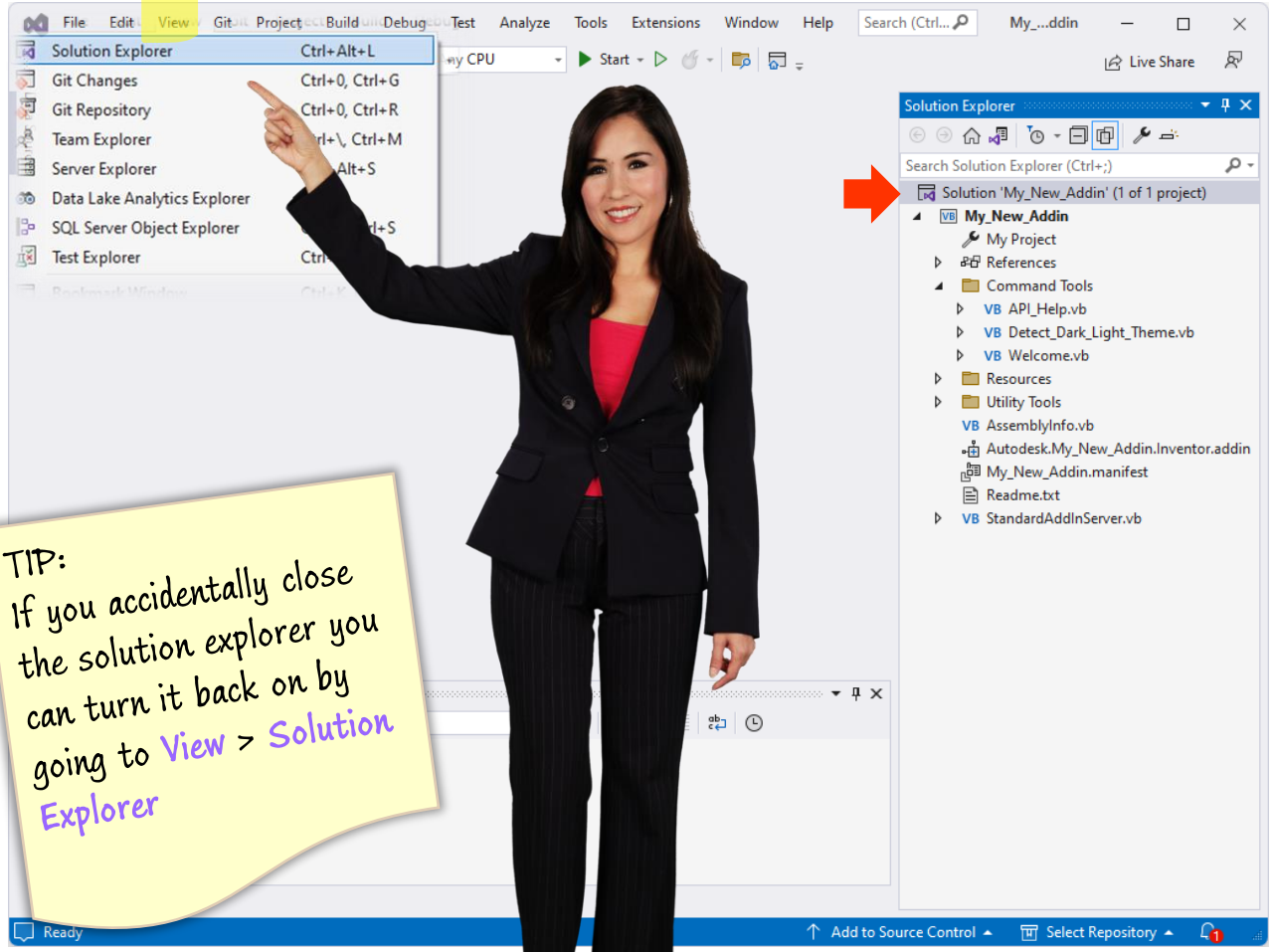
Creating an Add-in

Find the Inventor Add-in Templates

- Enter the project name
  - Do not use spaces!
  - Do not use dashes!
  - Spaces and dashes in the project name will cause the add-in not to load.
- Select the location where you want to store your add-in code files
- Optionally select the “Place solution and project in the same directory” checkbox
- Click Create to create the Project

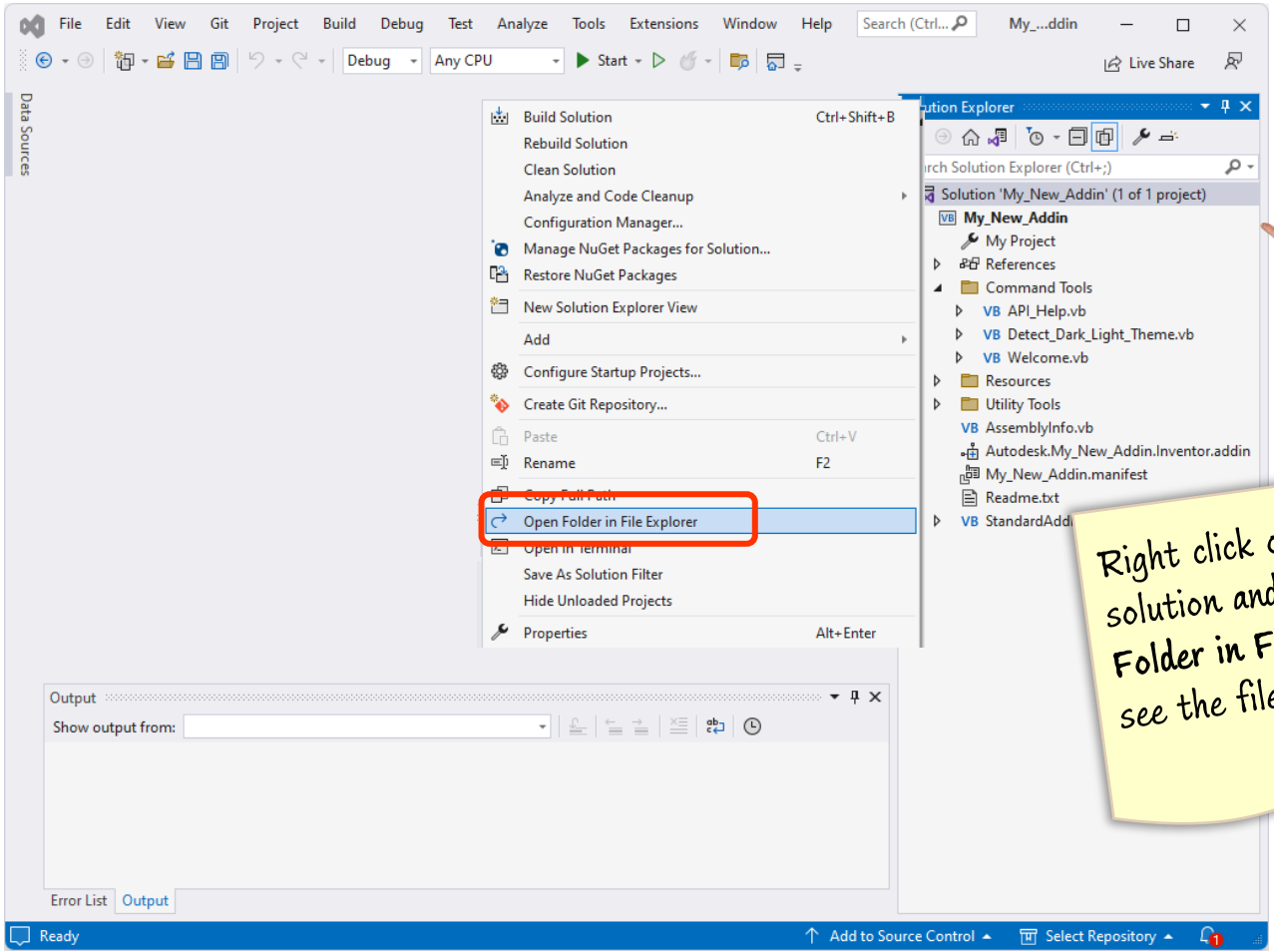


Creating an Add-in



- The solution is shown in the Solution Explorer

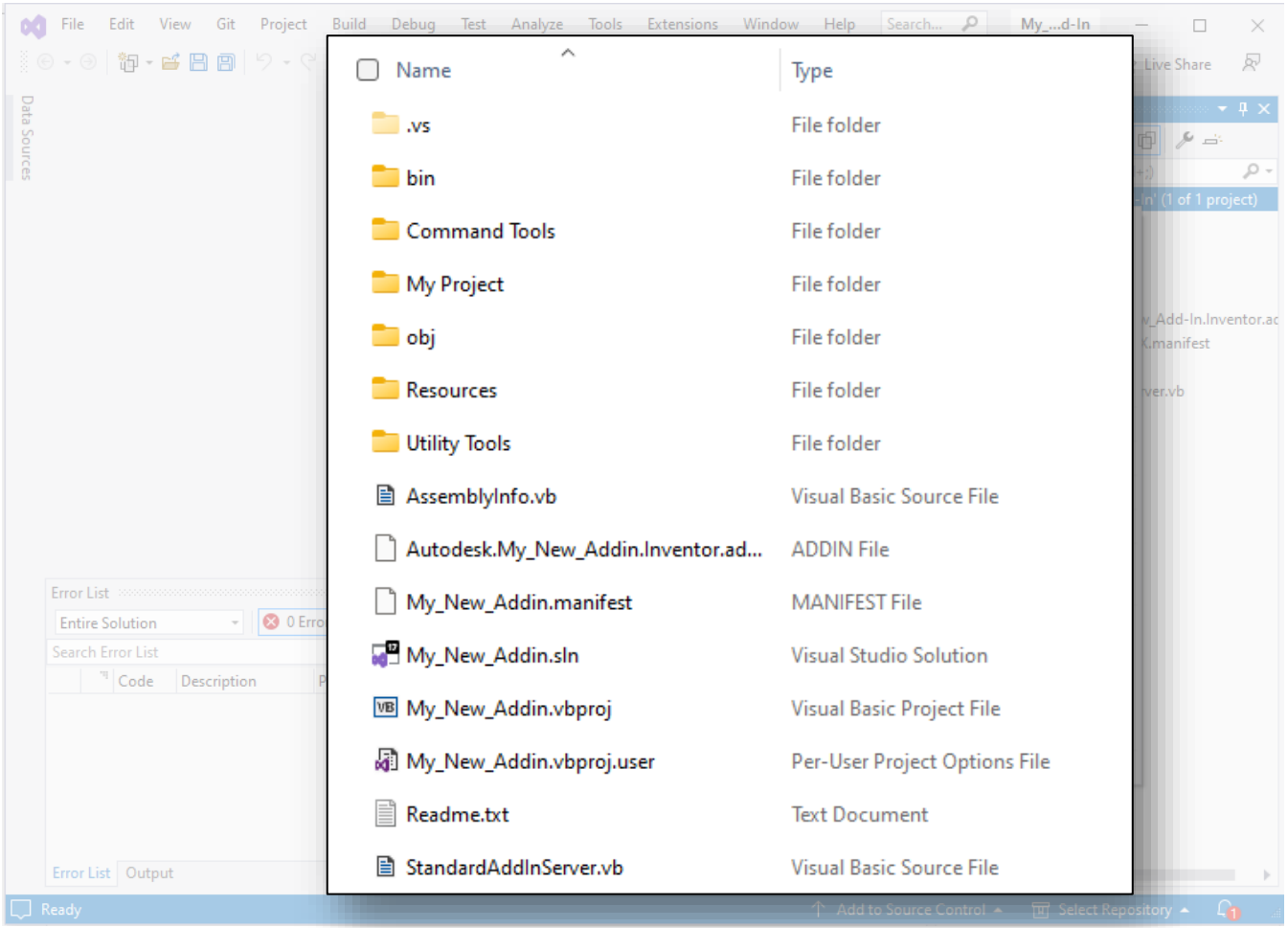
Creating an Add-in



- The solution is shown in the Solution Explorer

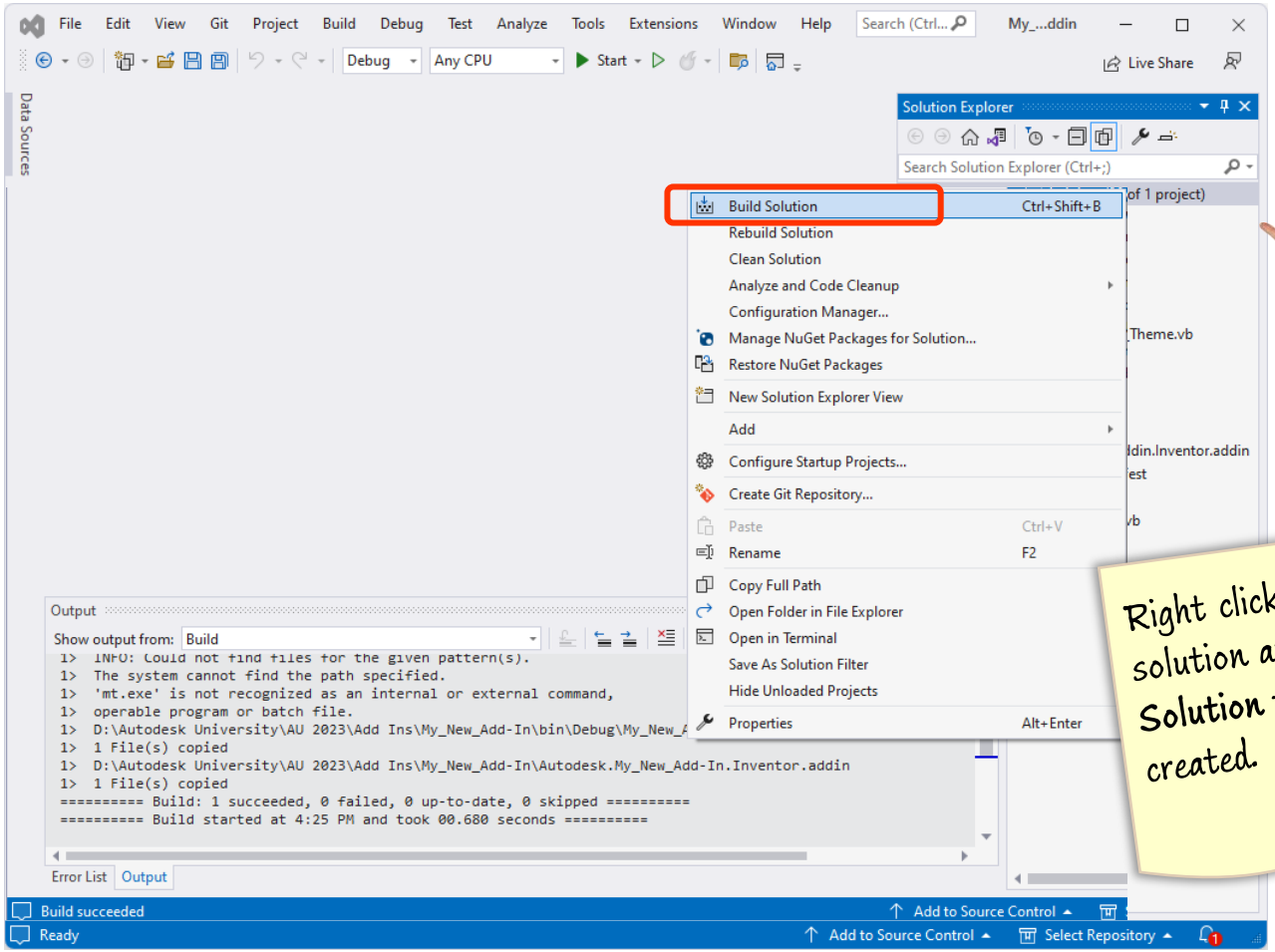
Right click on the solution and choose Open Folder in File Explorer to see the files created.

Creating an Add-in



- The folder will contain files and folders similar to image shown here

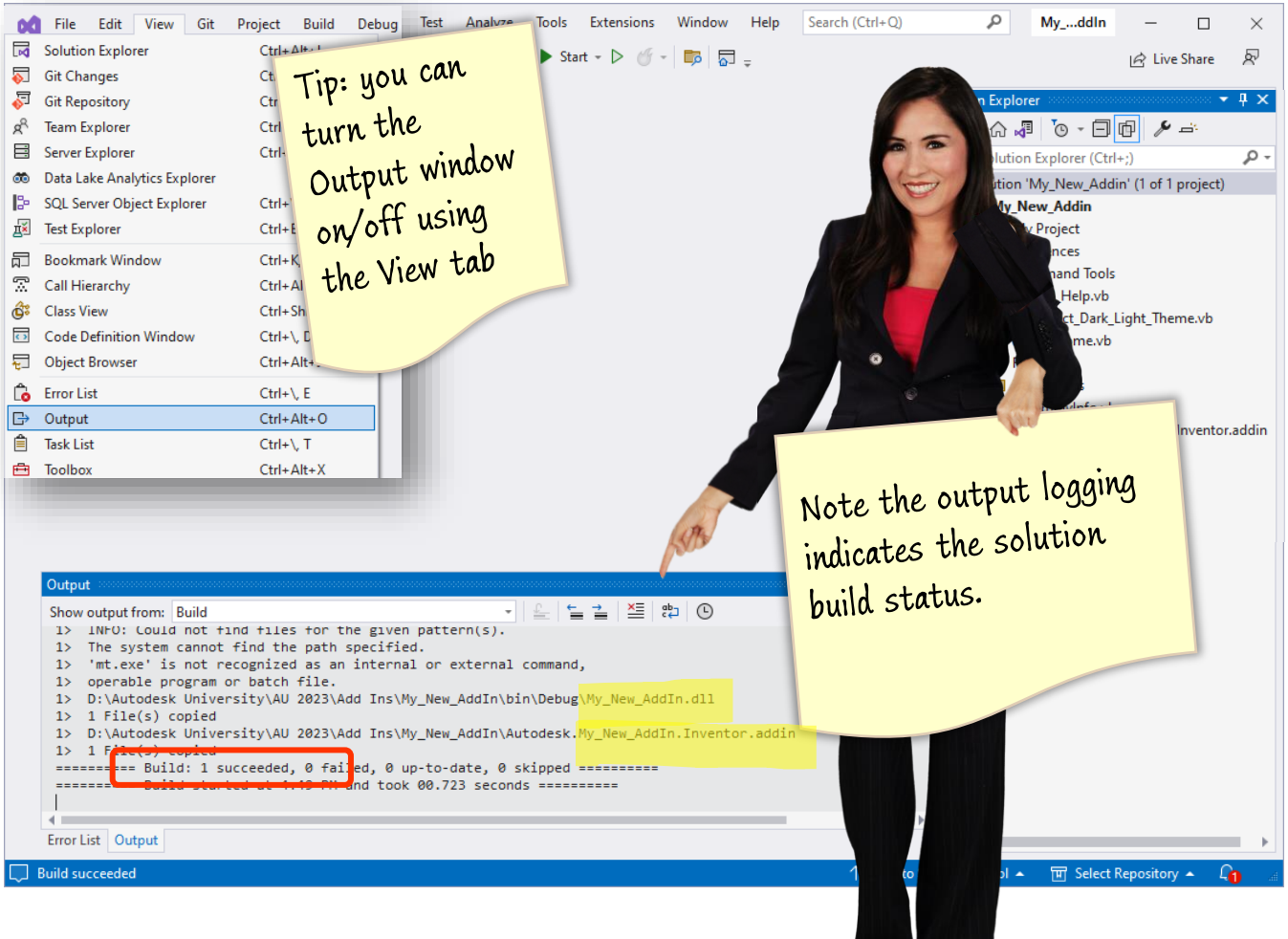
Creating an Add-in



- The solution is shown in the Solution Explorer

Right click on the solution and choose **Build Solution** to see the files created.

Creating an Add-in



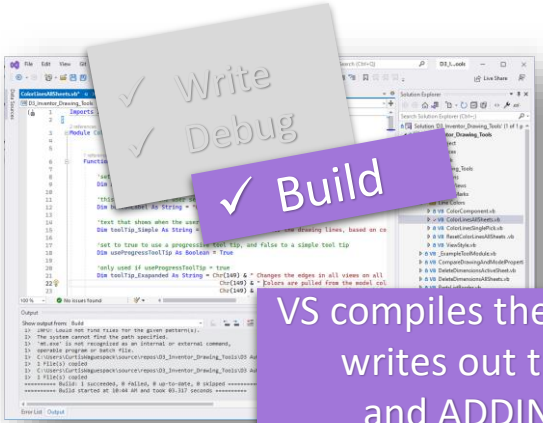
- The Build results are shown in the Output window

Overall Process

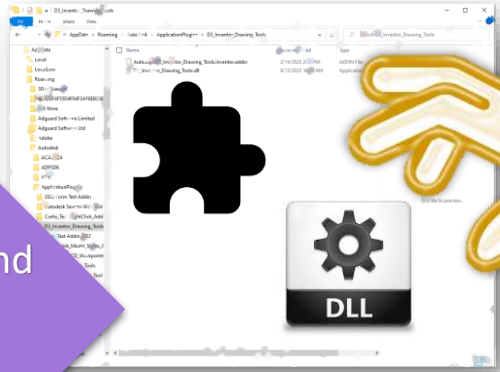
Visual Studio

Autodesk Application Plugins Folder

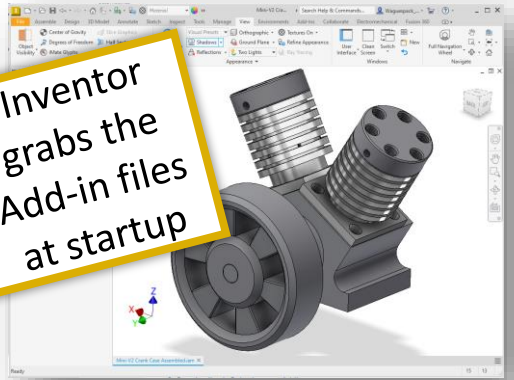
Autodesk Inventor



VS compiles the code and writes out the DLL and ADDIN files

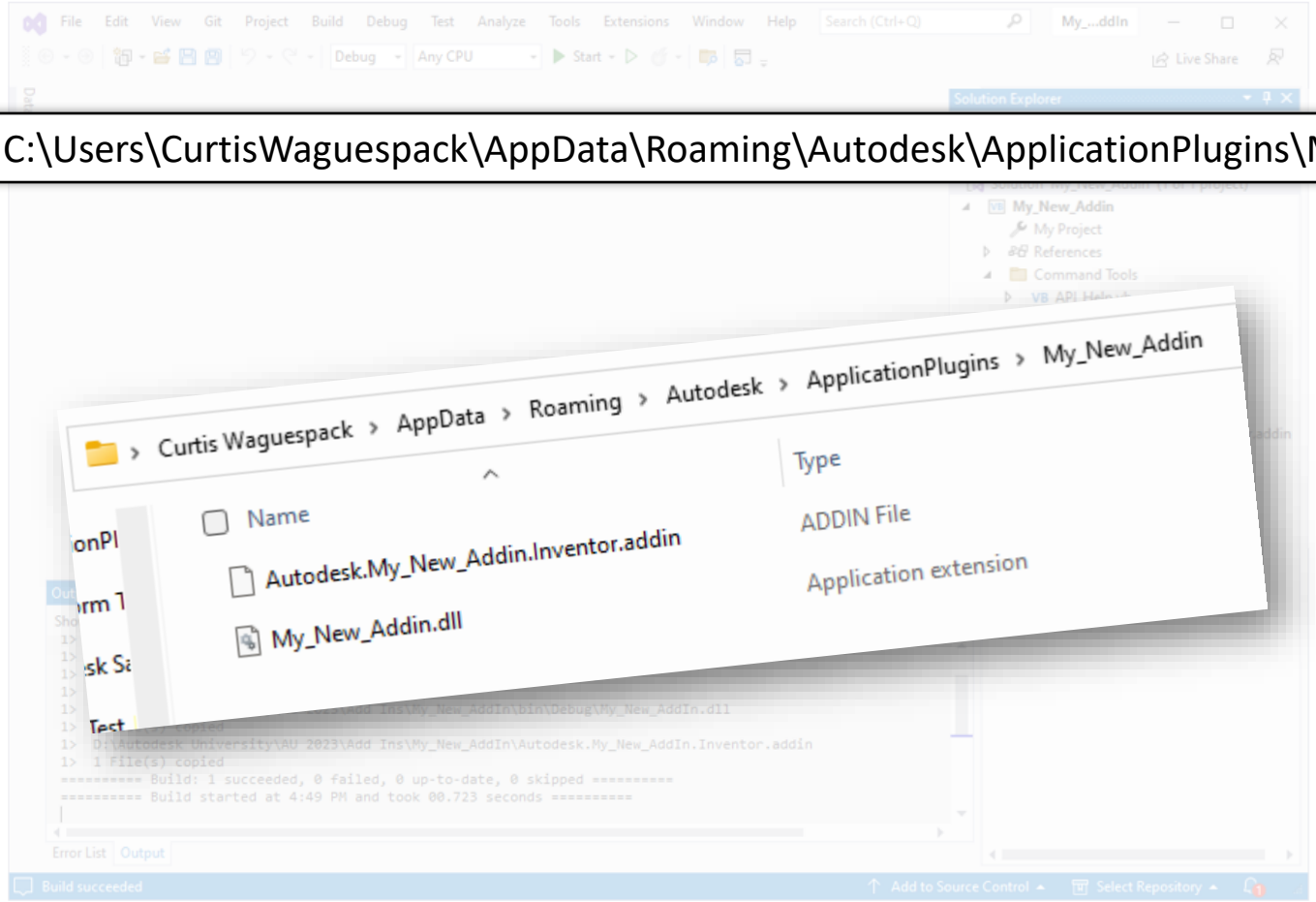


Inventor grabs the Add-in files at startup



Creating an Add-in

C:\Users\CurtisWaguespack\AppData\Roaming\Autodesk\ApplicationPlugins\My_New_Addin

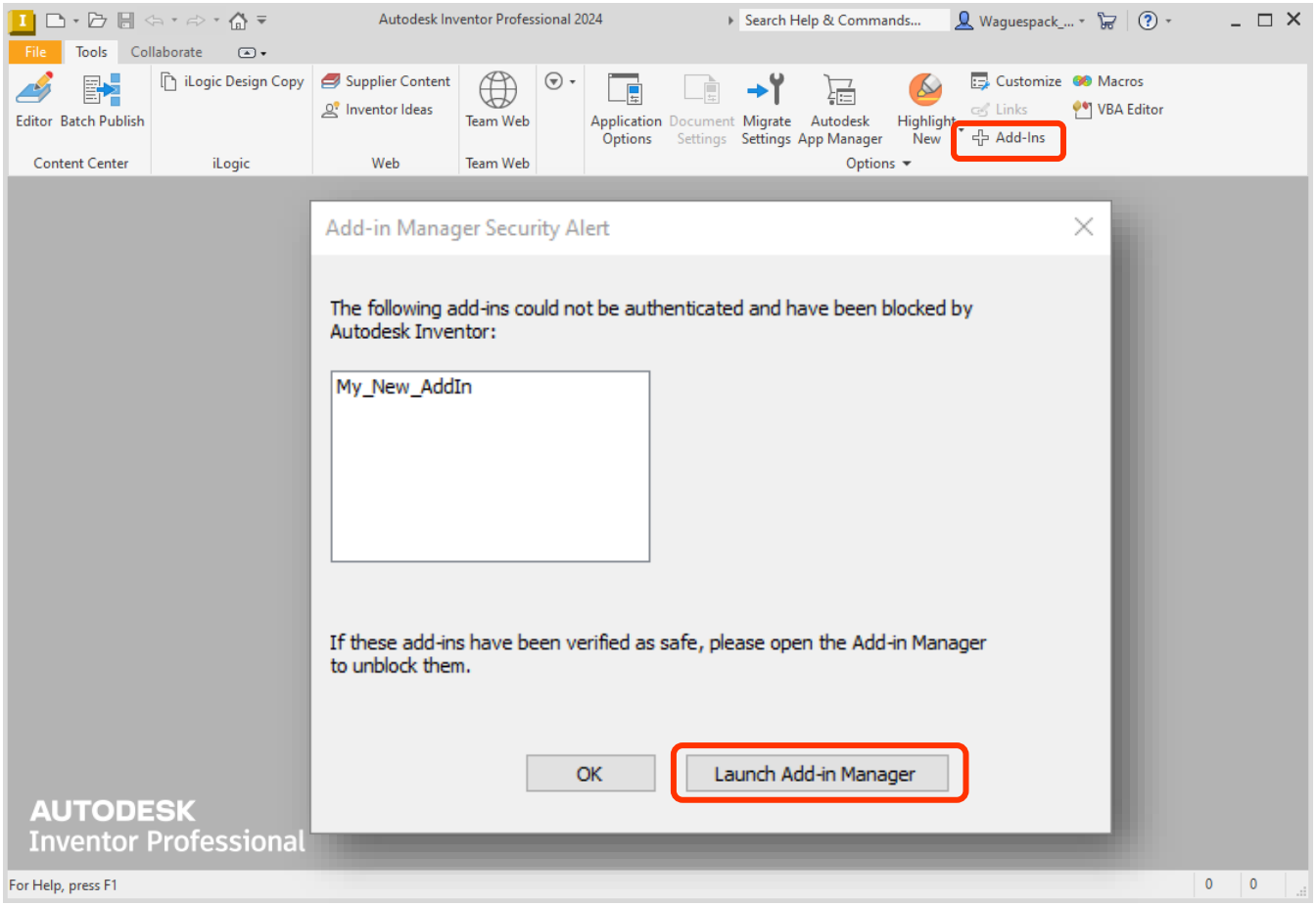


- Browse to the ApplicationPluggins folder and observe the files created during the Build

Loading the add-in

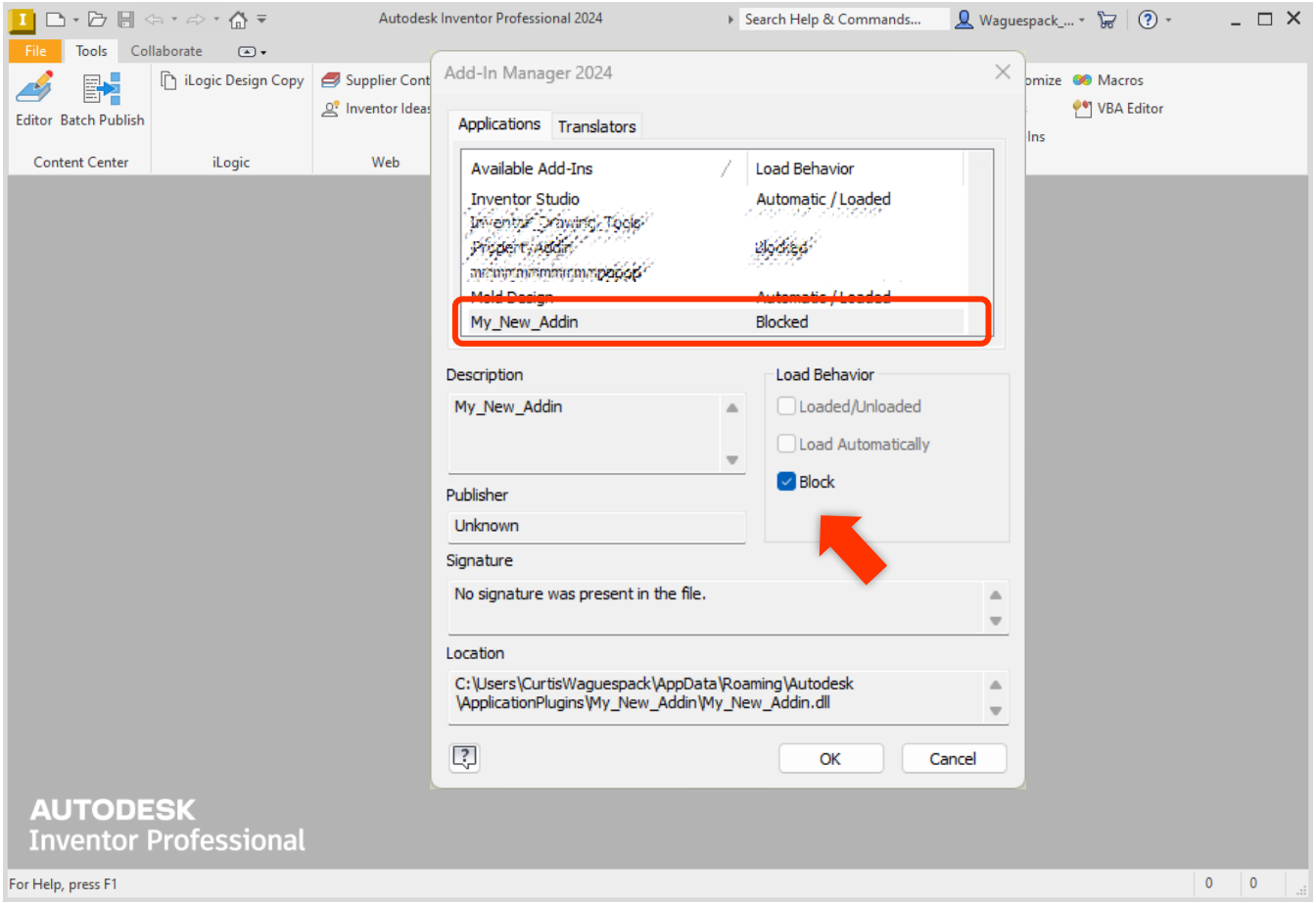
Working in Inventor

Loading an Add-in



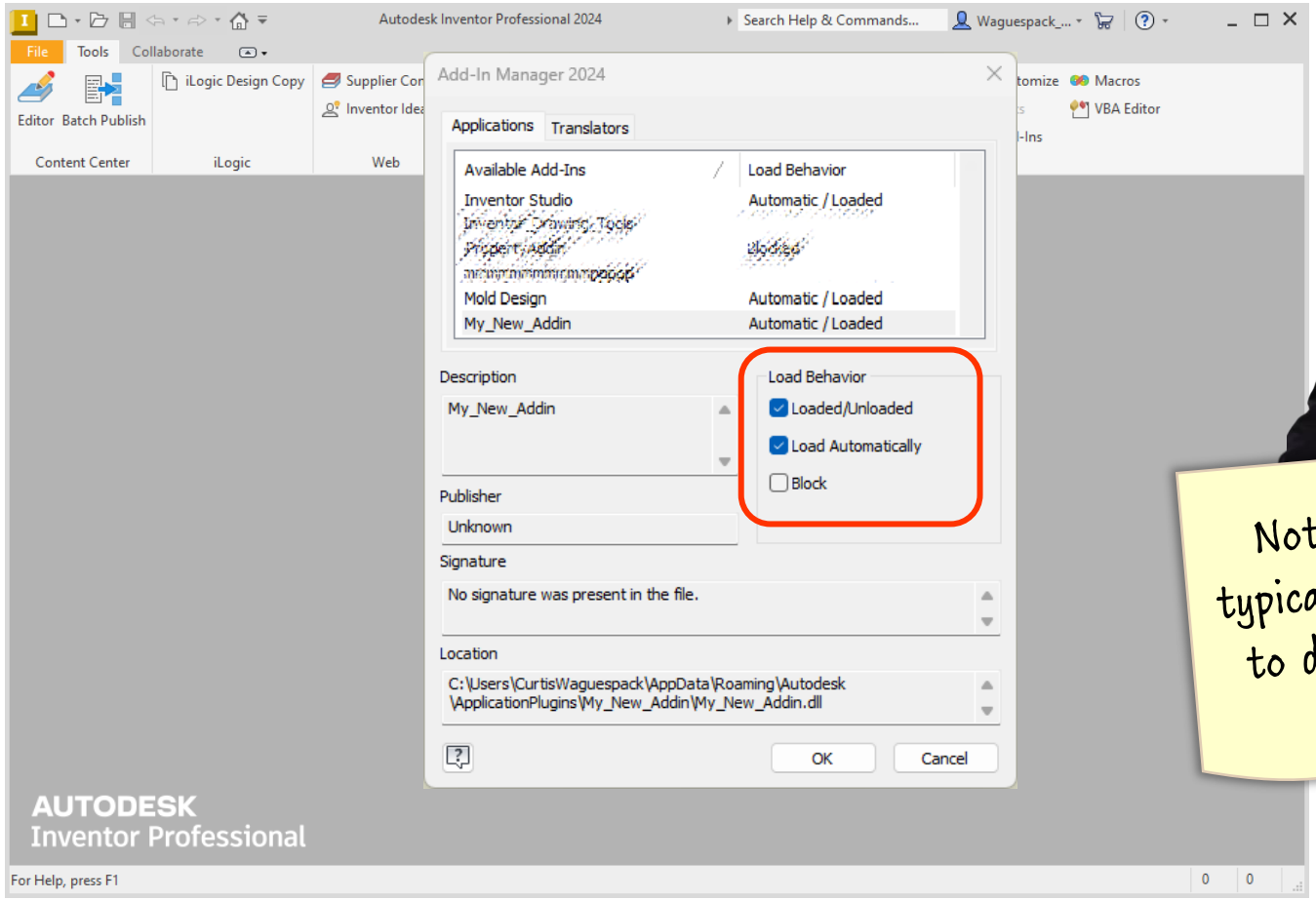
- Open Inventor.
- Note that you will be greeted by a message informing you that your add-in was blocked from loading
- Click the Launch Add-In Manager button or got to Tools > Add-ins

Loading an Add-in



- With the add-in selected, un-check the **Block** checkbox
- And then check the **Loaded/Unloaded**
- Ad the **Load Automatically** check box

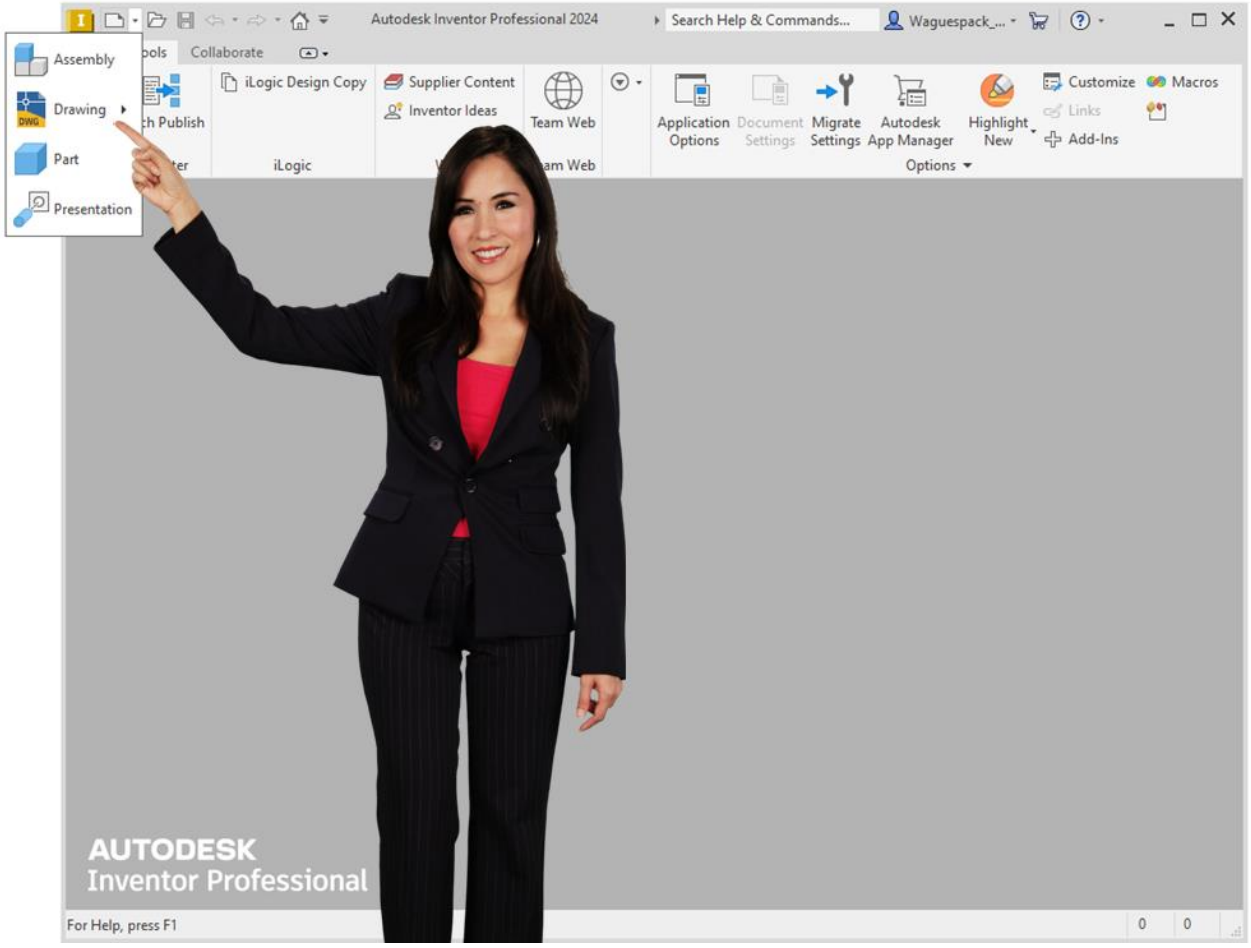
Loading an Add-in



Using the add-in

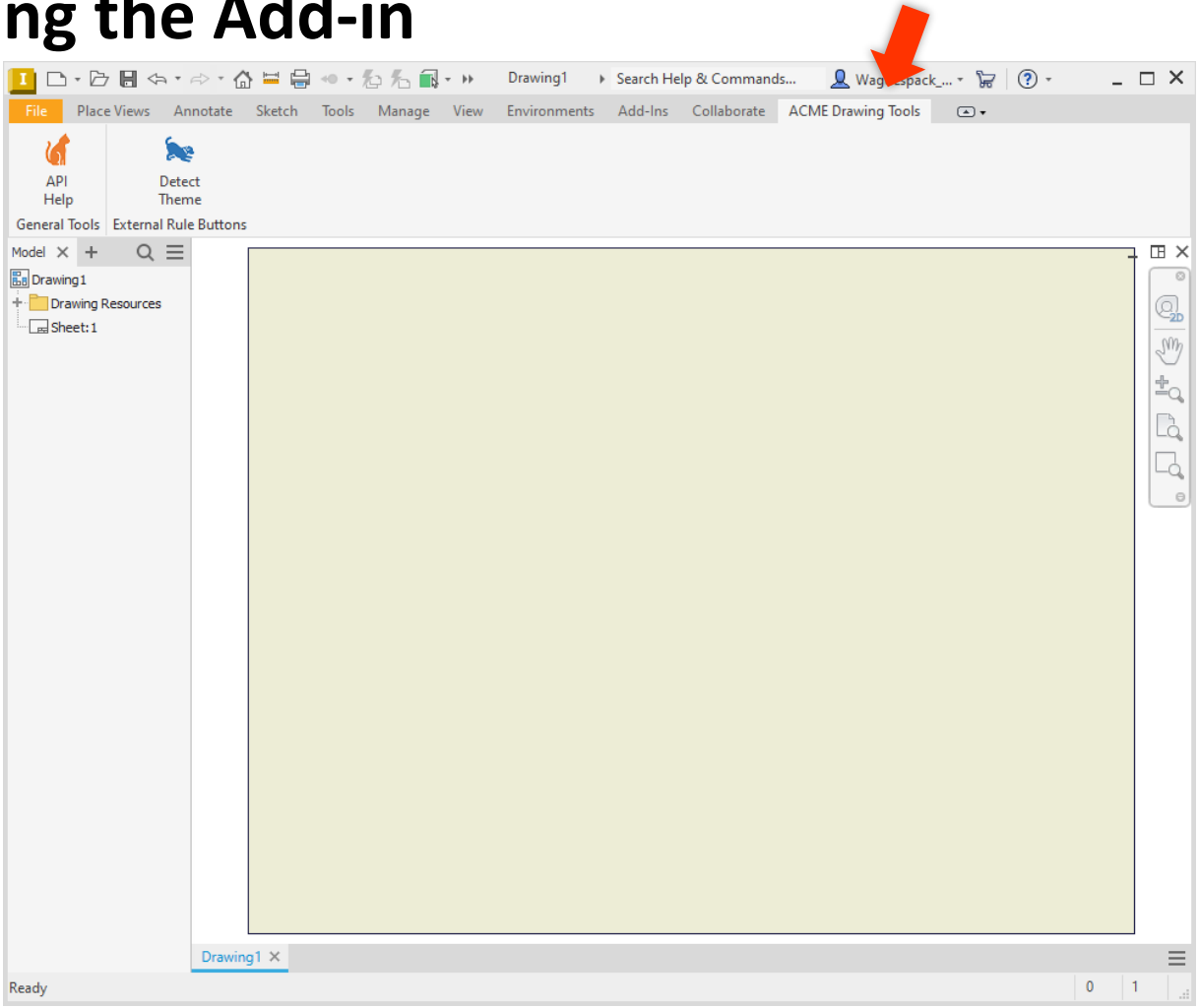
Working in Inventor

Using the Add-in



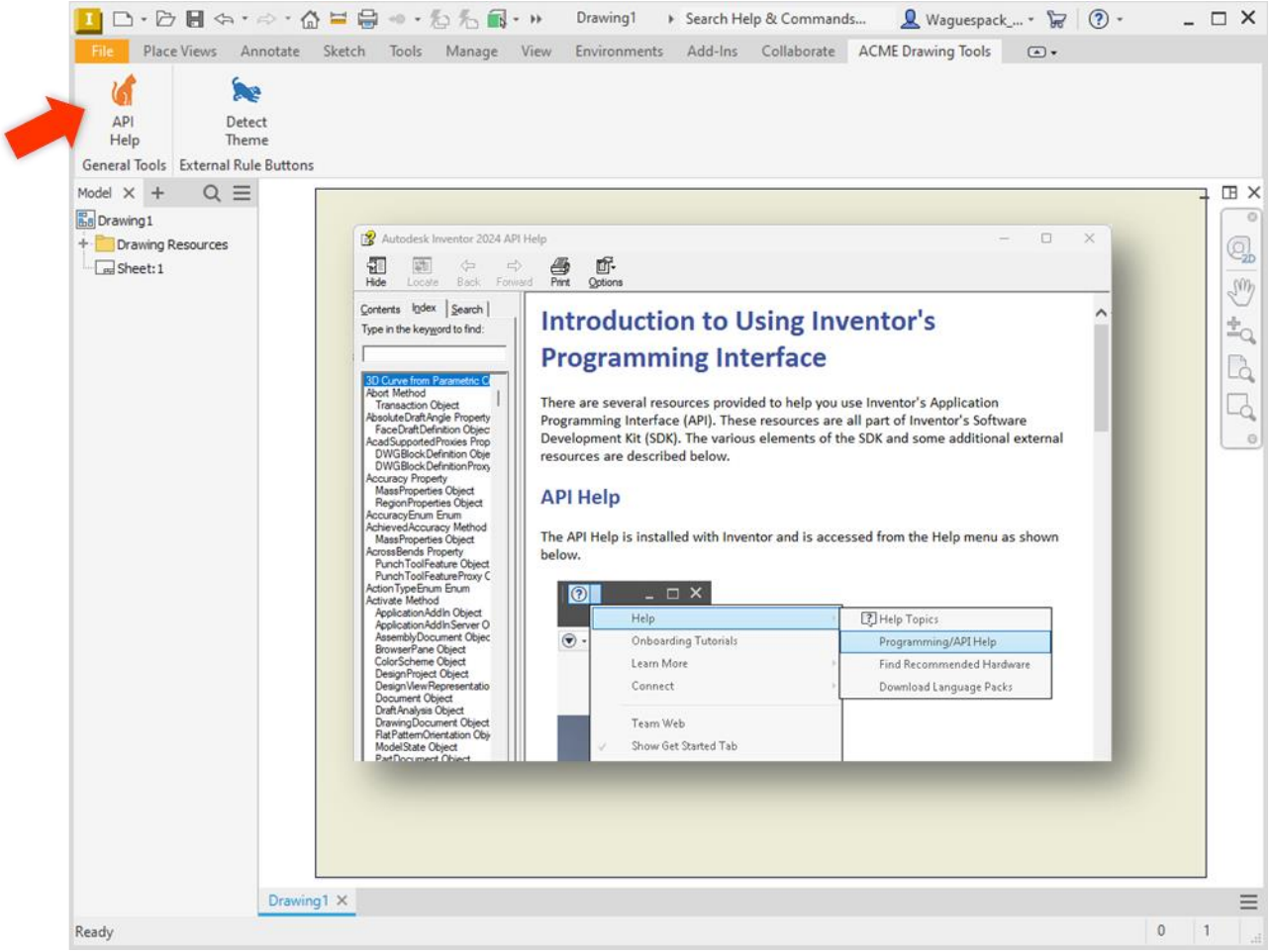
- Create a new drawing file from a template

Using the Add-in



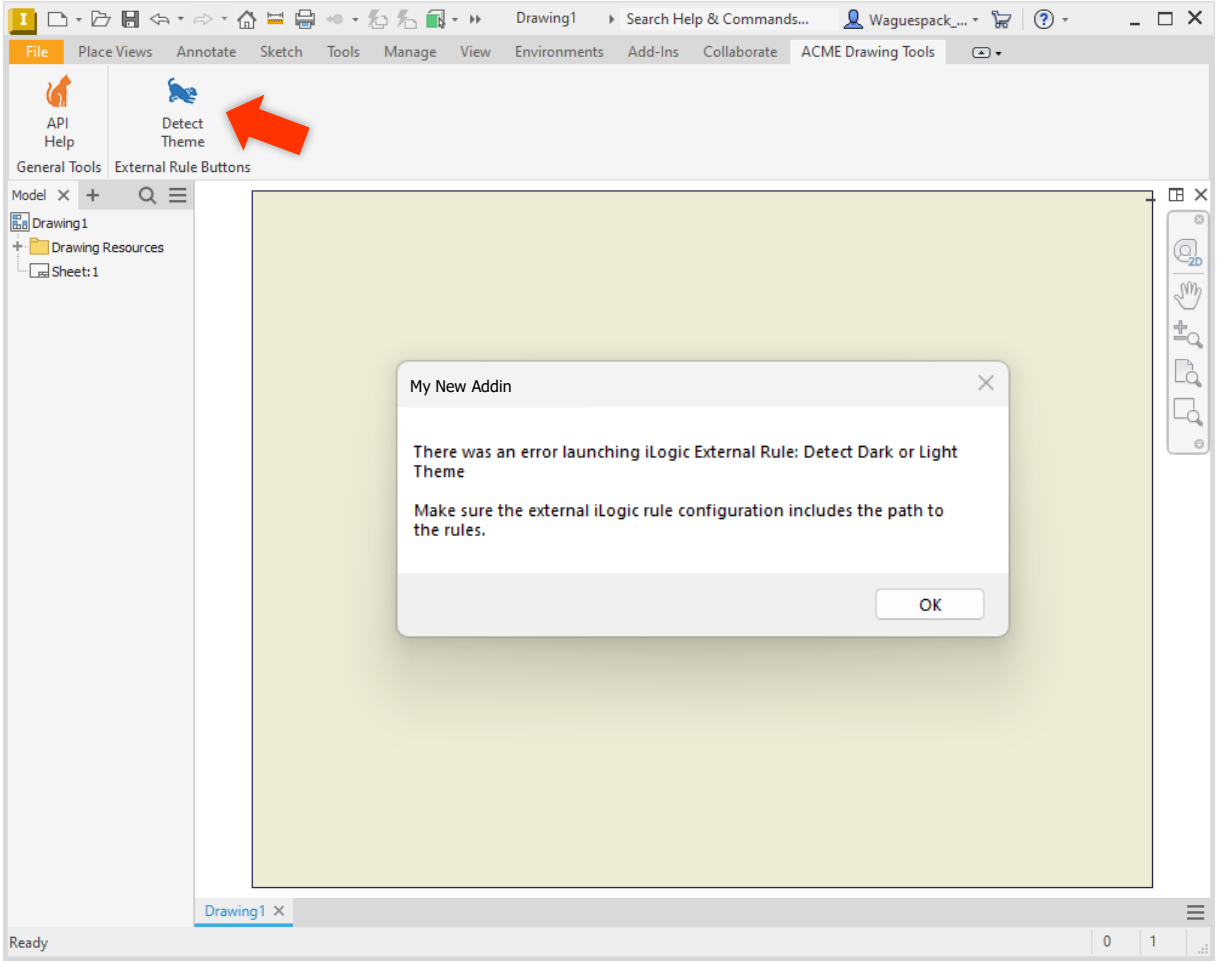
- Switch to the **ACME Drawing Tool** tab
- Note the 2 buttons found on this tab

Using the Add-in



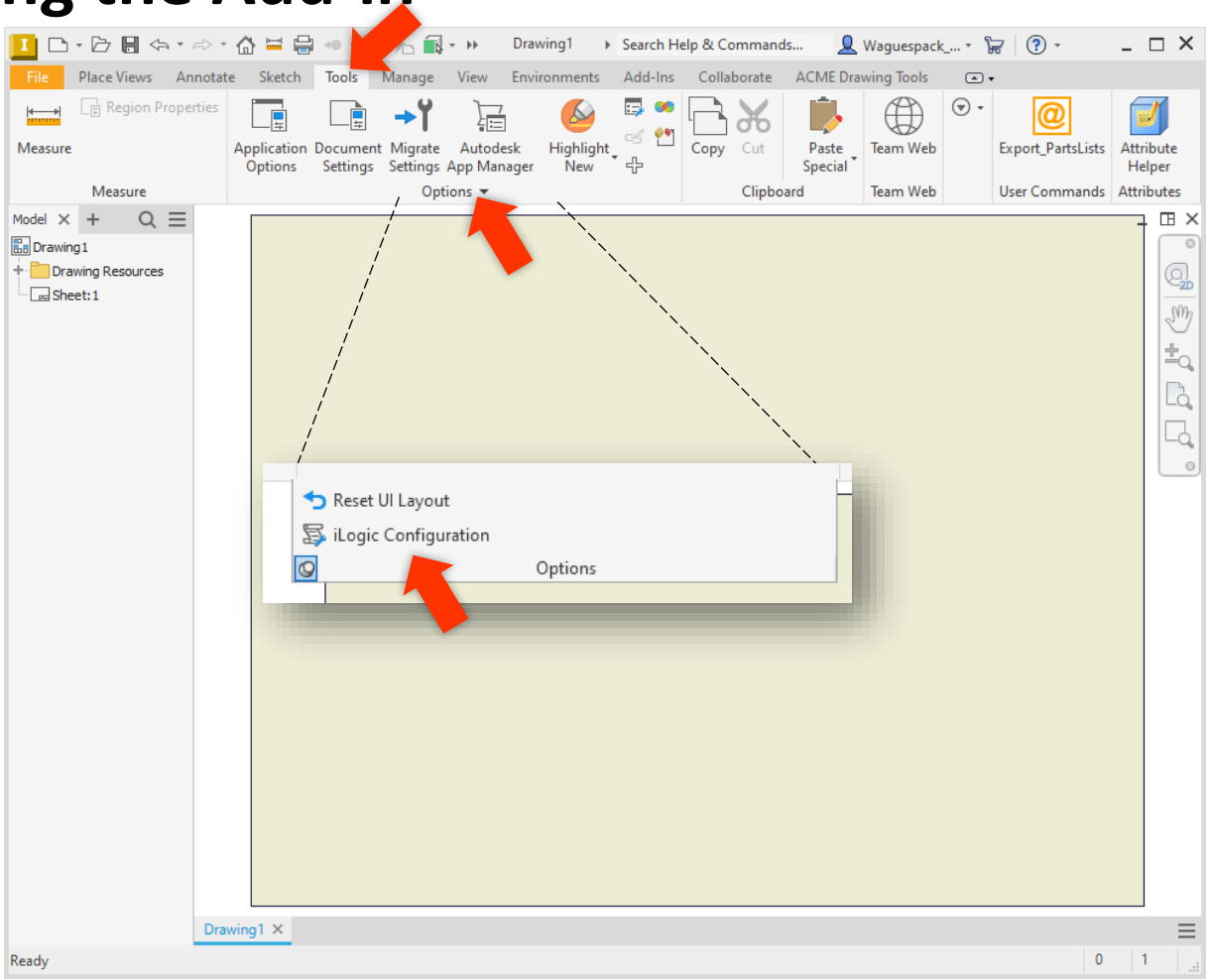
- Click the button called **API Help**
- This will open the Inventor API Help in a separate window
- This was done with code written in the add-in
- You can close the window if you like

Using the Add-in



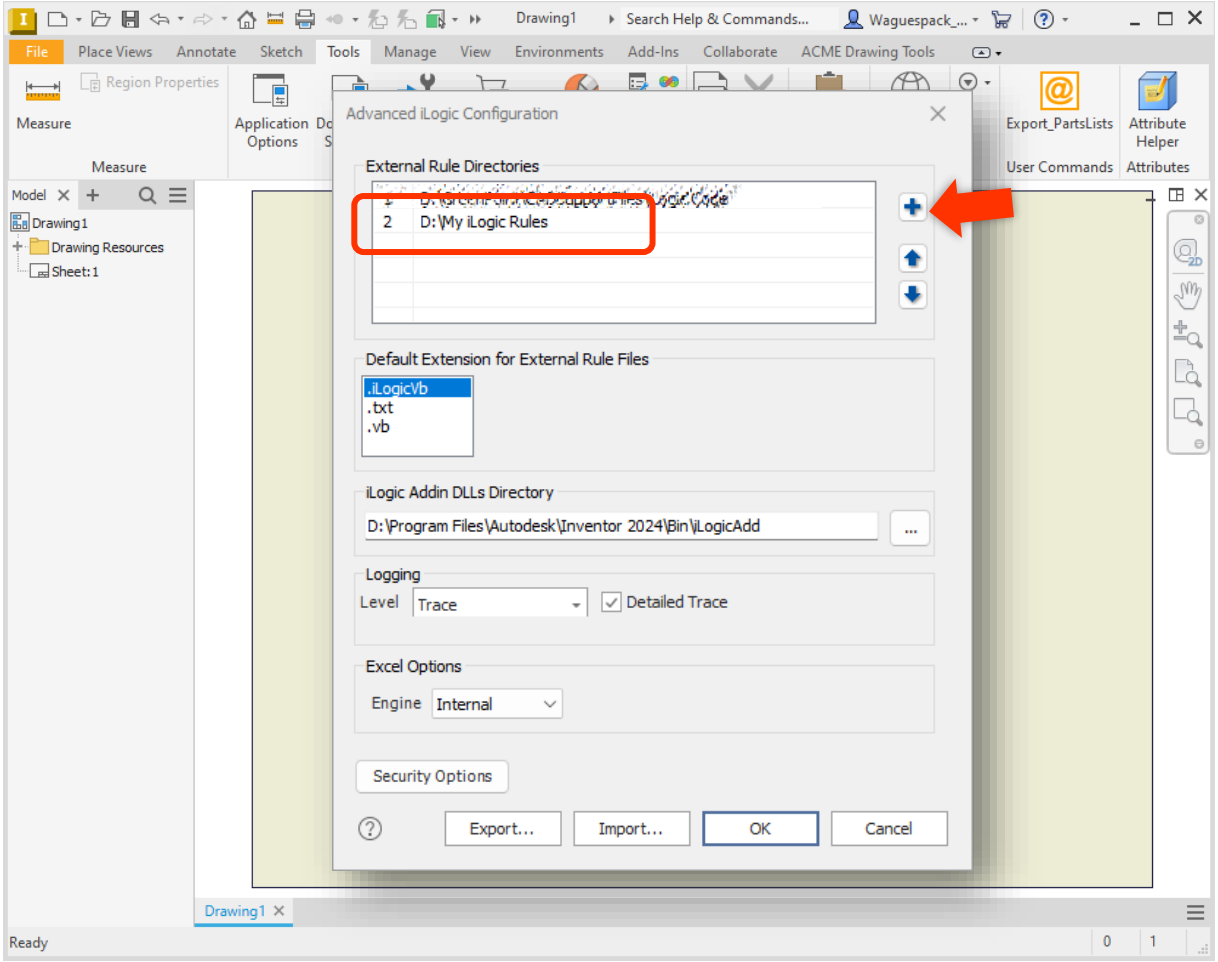
- Click the button called **API Help**
- This will open the Inventor API Help in a separate window
- This was done with code written in the add-in
- You can close the window if you like

Using the Add-in



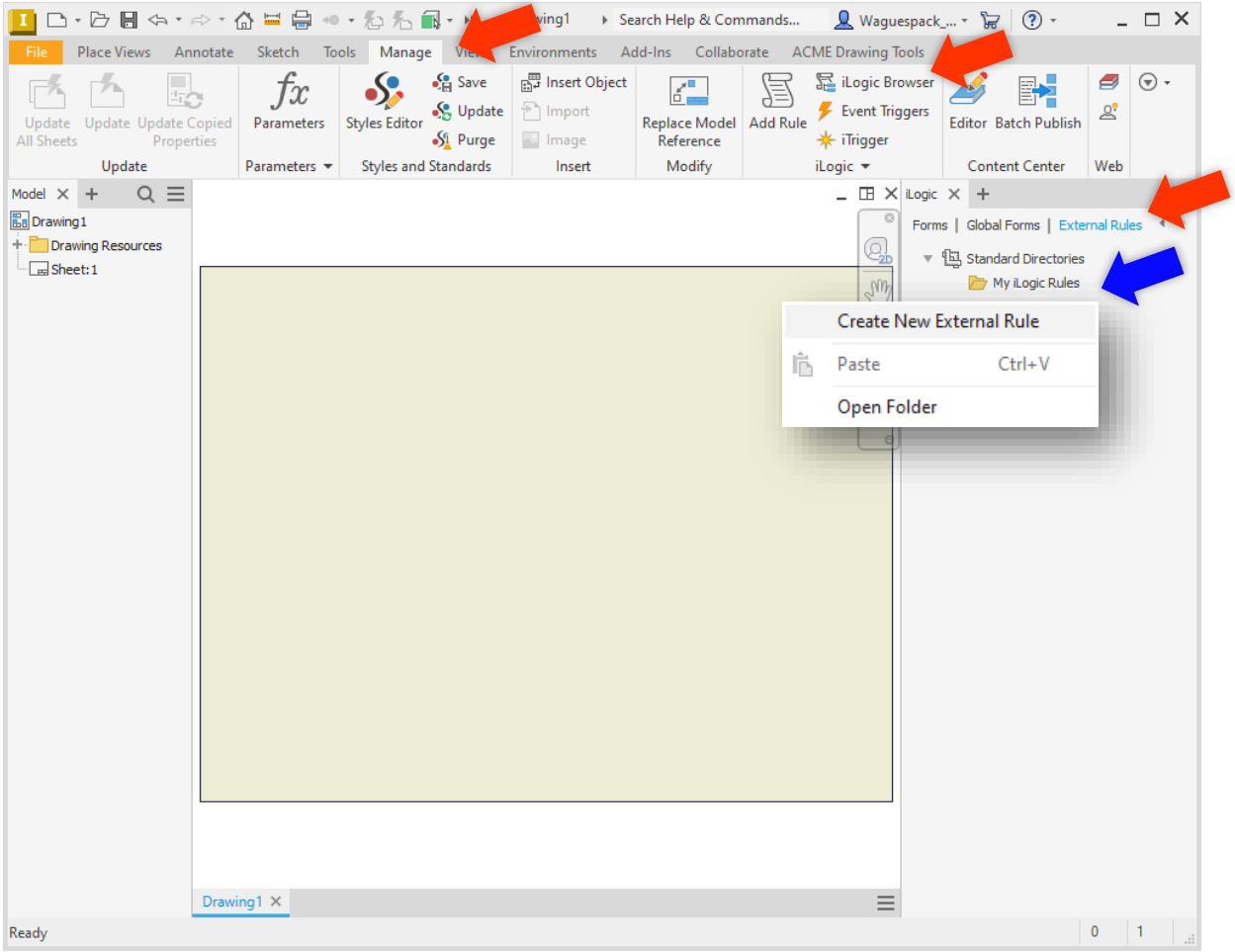
- Switch to the **Tools** tab and click the **Options** button
- Click the **iLogic Configuration** button as shown

Using the Add-in



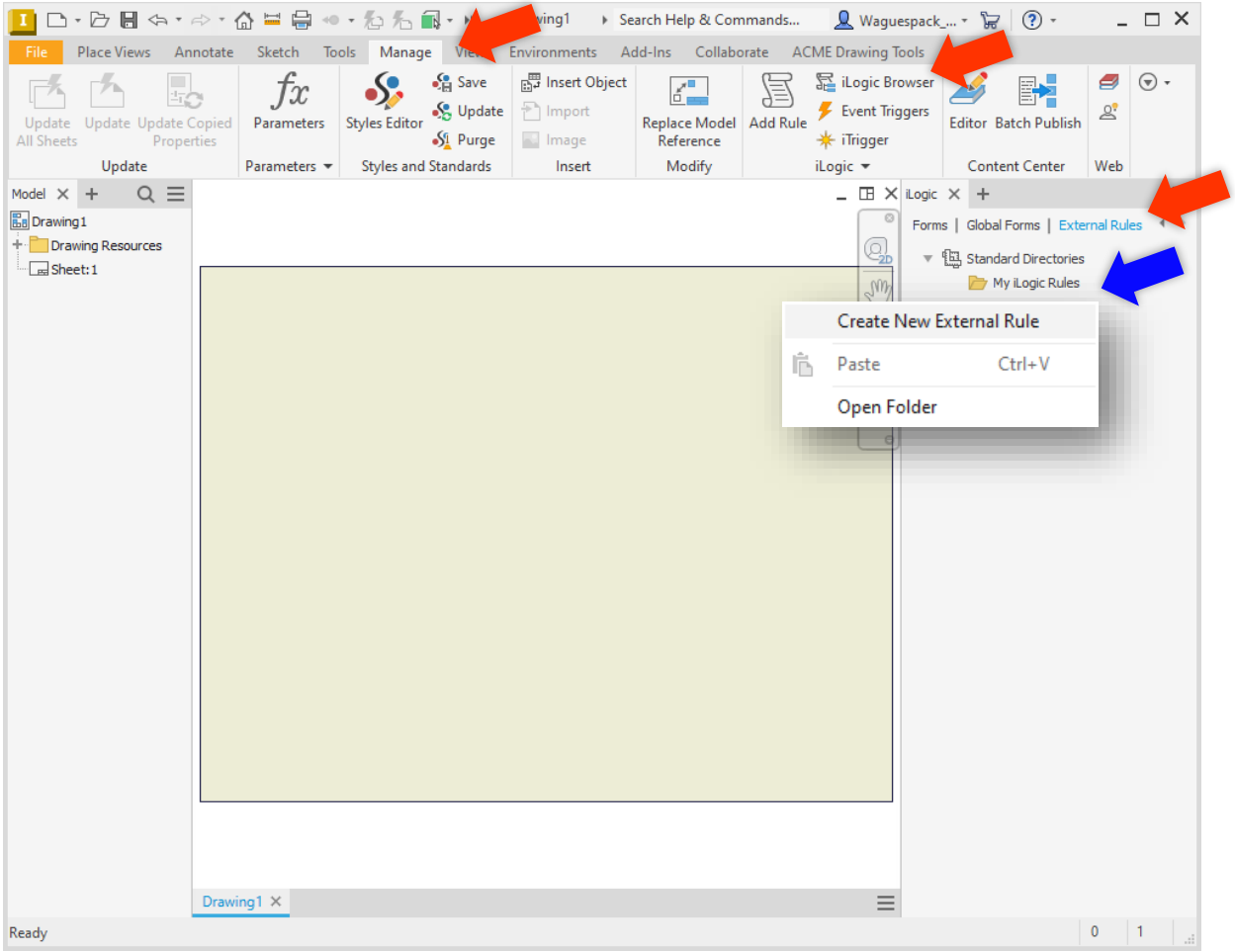
- If you have an External Rule directory already specified you can just use it
- If not use the **+** button to add a new path of your choice
- Click the **OK** button when done

Using the Add-in



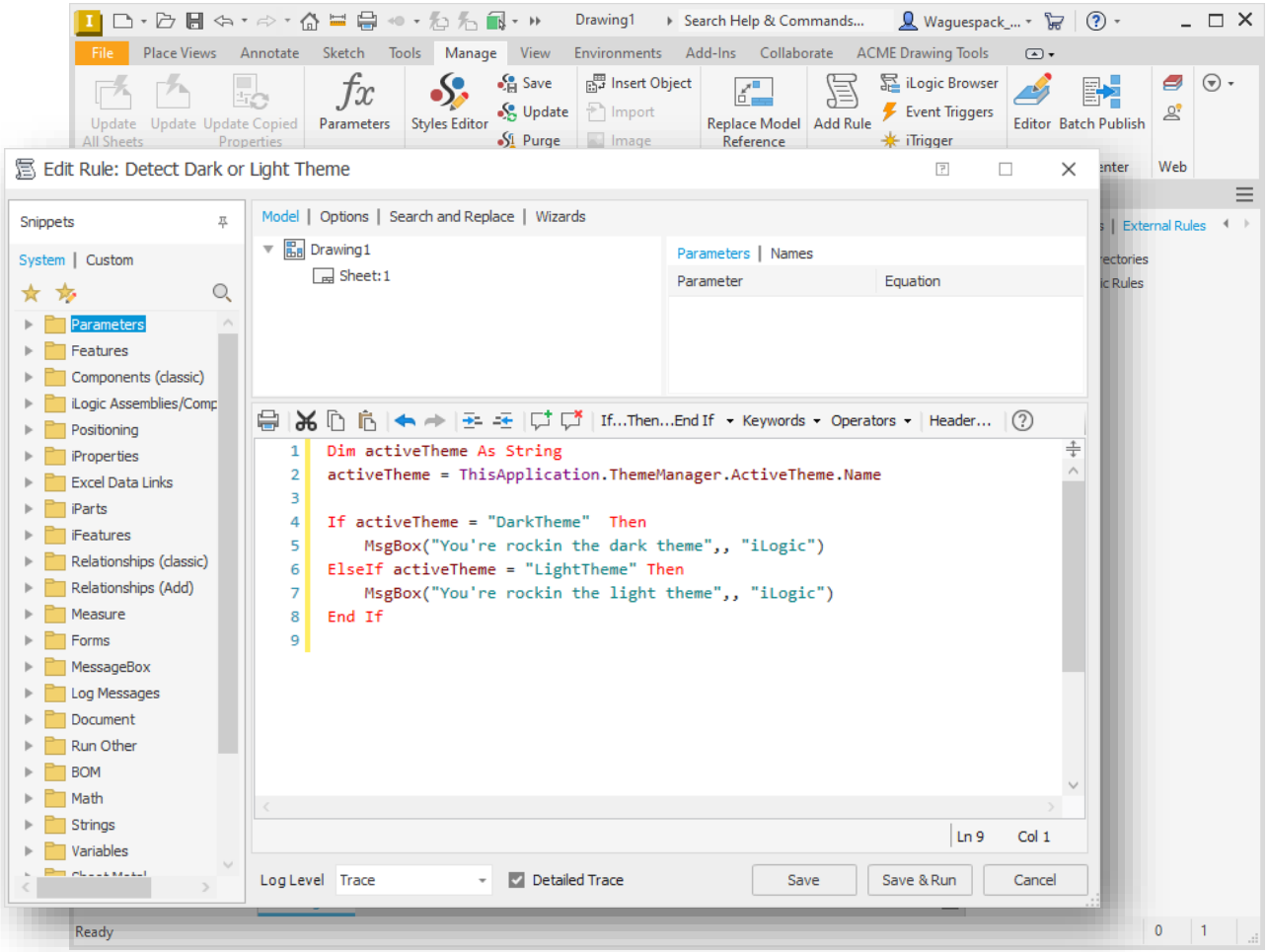
- On the **Manage** tab, click the **iLogic Browser** button
- Switch to the External Rules tab and note the folder you specified in the iLogic Configuration
- Right click on the folder and choose **Create New External Rule**

Using the Add-in



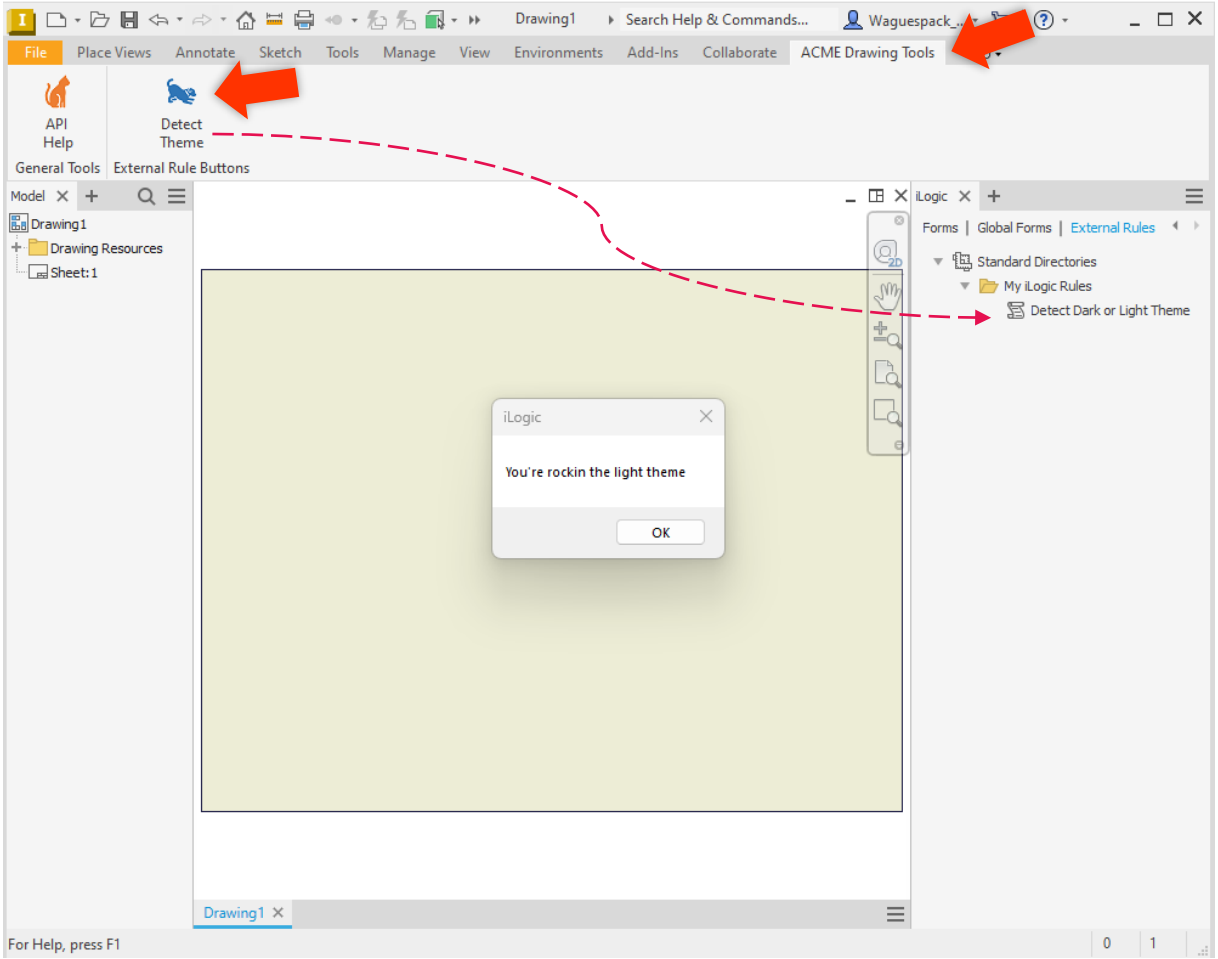
- On the **Manage** tab, click the **iLogic Browser** button
- Switch to the External Rules tab and note the folder you specified in the iLogic Configuration
- Right click on the folder and choose **Create New External Rule**

Using the Add-in



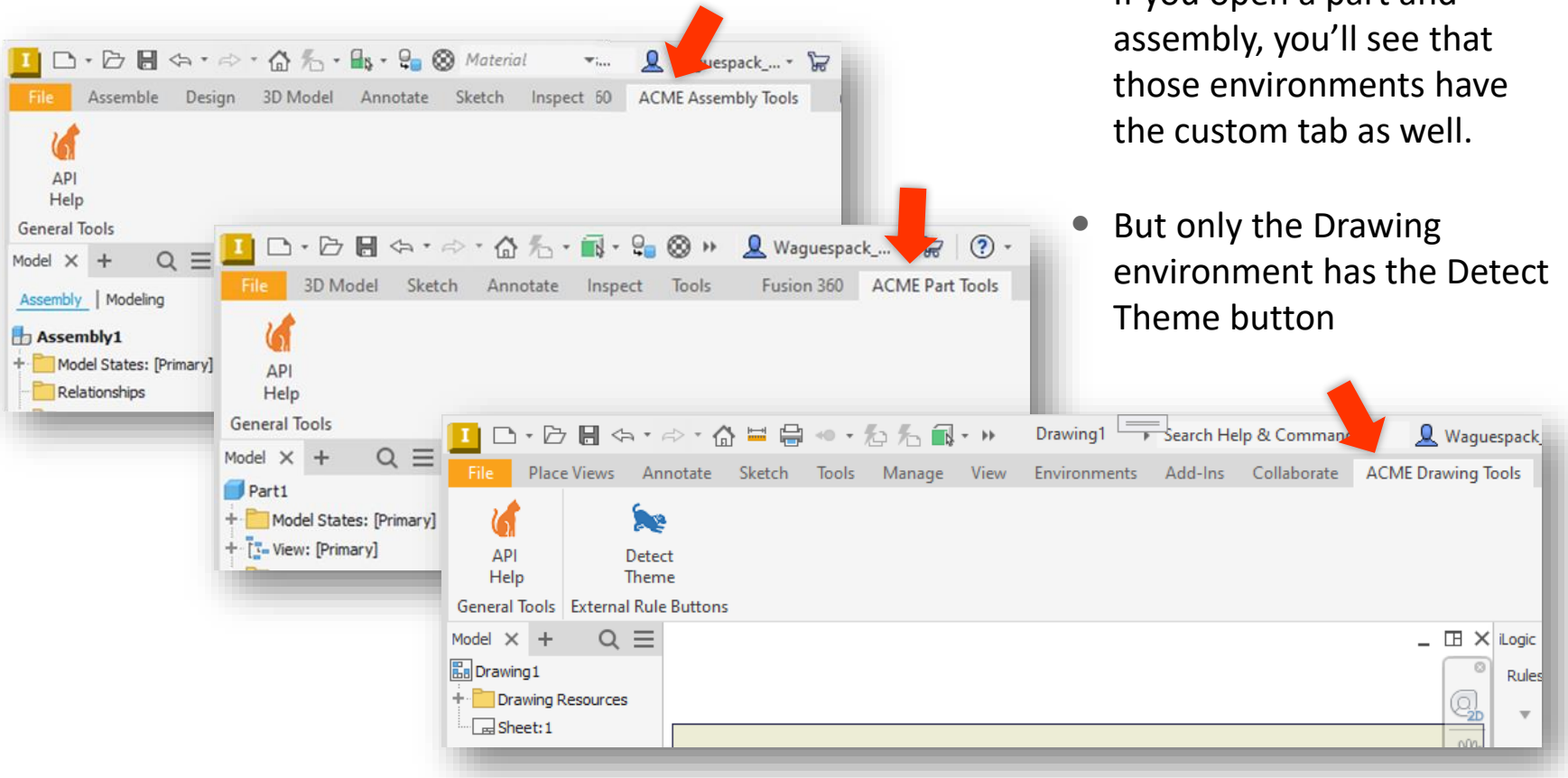
- Name the rule **Detect Dark or**
- Enter the code as shown
- Click the **Save** button, then the **Close** button
- Note that **this rule is available in the downloaded materials** if you'd prefer just to copy it.

Using the Add-in



- Switch back to the **ACME Drawing Tools** tab
- Click the **Detect Theme** button
- Note that the add-in button runs the external iLogic rule

Using the Add-in



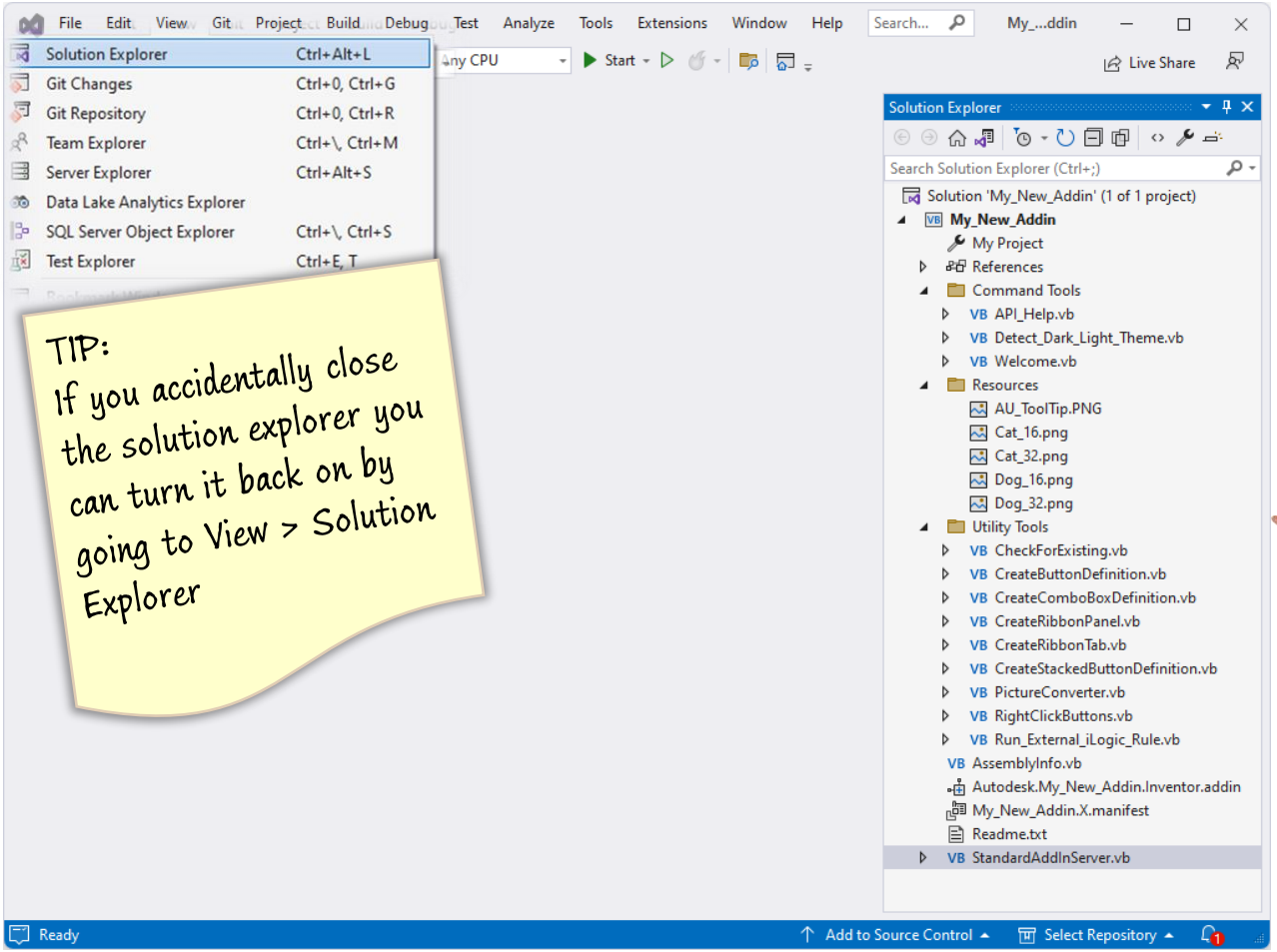
- If you open a part and assembly, you'll see that those environments have the custom tab as well.
- But only the Drawing environment has the Detect Theme button

AU 2023

Understanding the Add-in Template Structure

Working in Visual Studio

Understanding the Add-in Template Structure



TIP:
If you accidentally close the solution explorer you can turn it back on by going to View > Solution Explorer

- Back in Visual Studio, let's examine the file/folder structure of the template soluiton

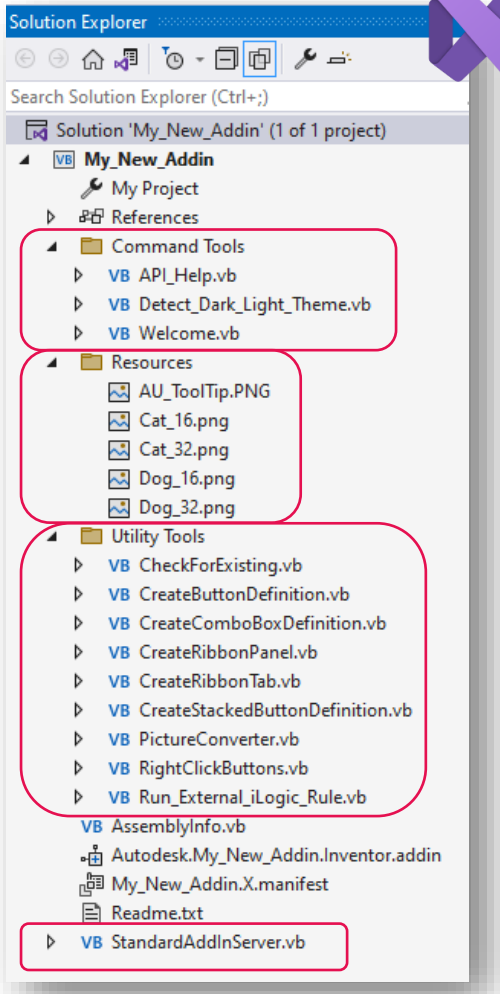
Understanding the Add-in Template Structure

Command Tool Modules →

Resources →

Utility Modules →

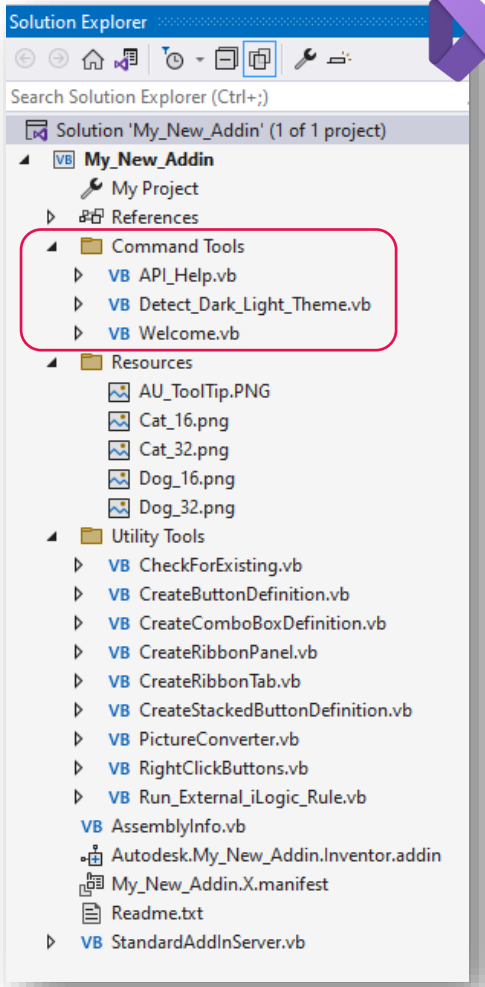
Standard Add-in Server →



Understanding the Add-in Template Structure

- One module for each button.
  - Function to define the button
  - Sub to house the code that is run when the button is clicked

Command Tool Modules →



Understanding the Add-in Template Structure

API_Help.vb* | My_New_Addin | API_Help | CreateButton

```
1   Imports Inventor
2
    4 references
3   Module API_Help
4
5       ''' <summary> Creates a button Button Definition Define the label, icons, tool t ...
13
14      'This function is where the button is defined
        3 references
15      Function CreateButton(environment As String, CustomDrawingTab As RibbonTab, ribbonPanel As RibbonPanel,
16                            useLargeIcon As Boolean, isInButtonStack As Boolean) As ButtonDefinition
17
18          'get the images to use for the button
19          Dim largeIcon As IPictureDisp = PictureDispConverter.ToIPictureDisp(My.Resources.Cat_32)
20          Dim standardIcon As IPictureDisp = PictureDispConverter.ToIPictureDisp(My.Resources.Cat_16)
21          Dim toolTipImage As IPictureDisp = Nothing
22
23          'this is the text the user sees on the button
24          Dim buttonLabel As String = "API " & vbLf & "Help"
25
26          'text that displays when the user hovers over the button
27          Dim toolTip_Simple As String = "Opens the API Help"
28          Dim toolTip_Expanded As String = Nothing
29
30
31      Progressive ToolTip
47
48          Dim buttonDef As ButtonDefinition
49          buttonDef = CreateButtonDefintion.CreateButtonDef(environment, CustomDrawingTab, ribbonPanel, useLa
50                                                            isInButtonStack, useProgressToolTip,
51                                                            buttonLabel, toolTip_Simple, toolTip_Expanc
52                                                            standardIcon, largeIcon, toolTipImage)
53
54          Return buttonDef
55
56      End Function
77
78  End Module
79
```

- One module for each button.
  - Function to define the button
  - Sub to house the code that is run when the button is clicked

Understanding the Add-in Template Structure

API_Help.vb* | My_New_Addin | API_Help | CreateButton

```vb
Imports Inventor

4 references
Module API_Help

    ''' <summary> Creates a button Button Definition Define the label, icons, tool t ...

    'This function is where the button is defined
    3 references
    Function CreateButton(environment As String, CustomDrawingTab As RibbonTab, ribbonPanel As RibbonPanel,
                          useLargeIcon As Boolean, isInButtonStack As Boolean) As ButtonDefinition

        'get the images to use for the button
        Dim largeIcon As IPictureDisp = PictureDispConverter.ToIPictureDisp(My.Resources.Cat_32)
        Dim standardIcon As IPictureDisp = PictureDispConverter.ToIPictureDisp(My.Resources.Cat_16)
        Dim toolTipImage As IPictureDisp = Nothing

        'this is the text the user sees on the button
        Dim buttonLabel As String = "API " & vbLf & "Help"

        'text that displays when the user hovers over the button
        Dim toolTip_Simple As String = "Opens the API Help"
        Dim toolTip_Expanded As String = Nothing


    Progressive ToolTip

        Dim buttonDef As ButtonDefinition
        buttonDef = CreateButtonDefintion.CreateButtonDef(environment, CustomDrawingTab, ribbonPanel, useLargeIcon,
                                                          isInButtonStack, useProgressToolTip,
                                                          buttonLabel, toolTip_Simple, toolTip_Expanded,
                                                          standardIcon, largeIcon, toolTipImage)

        Return buttonDef

    End Function


End Module
```

- One module for each button.
  - Function to define the button
  - Sub to house the code that is run when the button is clicked

Understanding the Add-in Template Structure

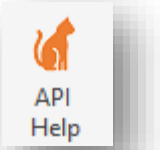
API_Help.vb*

My_New_Addin | API_Help | CreateButton

```
1   Imports Inventor
2
    4 references
3   Module API_Help
4
5       ''' <summary> Creates a button Button Definition Define the label, icons, tool t ...
13
14      'This function is where the button is defined
        3 references
15      Function CreateButton(environment As String, CustomDrawingTab As RibbonTab, ribbonPanel As RibbonPanel,
16                            useLargeIcon As Boolean, isInButtonStack As Boolean) As ButtonDefinition
17
18          'get the images to use for the button
19          Dim largeIcon As IPictureDisp = PictureDispConverter.ToIPictureDisp(My.Resources.Cat_32)
20          Dim standardIcon As IPictureDisp = PictureDispConverter.ToIPictureDisp(My.Resources.Cat_16)
21          Dim toolTipImage As IPictureDisp = Nothing
22
23          'this is the text the user sees on the button
24          Dim buttonLabel As String = "API " & vbLf & "Help"
25
26          'text that displays when the user hovers over the button
27          Dim toolTip_Simple As String = "Opens the API Help"
28          Dim toolTip_Expanded As String = Nothing
29
30
31      Progressive ToolTip
47
48          Dim buttonDef As ButtonDefinition
49          buttonDef = CreateButtonDefintion.CreateButtonDef(environment, CustomDrawingTab, ribbonPanel, useLargeIcon,
50                                                            isInButtonStack, useProgressToolTip,
51                                                            buttonLabel, toolTip_Simple, toolTip_Expanded,
52                                                            standardIcon, largeIcon, toolTipImage)
53
54          Return buttonDef
55
56      End Function


77
78  End Module
79
```

Specify the icons



Understanding the Add-in Template Structure

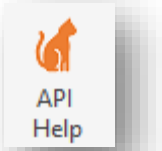
API_Help.vb*

My_New_Addin | API_Help | CreateButton

```
1   Imports Inventor
2
    4 references
3   Module API_Help
4
5       ''' <summary> Creates a button Button Definition Define the label, icons, tool t ...
13
14      'This function is where the button is defined
        3 references
15      Function CreateButton(environment As String, CustomDrawingTab As RibbonTab, ribbonPanel As RibbonPanel,
16                            useLargeIcon As Boolean, isInButtonStack As Boolean) As ButtonDefinition
17
18          'get the images to use for the button
19          Dim largeIcon As IPictureDisp = PictureDispConverter.ToIPictureDisp(My.Resources.Cat_32)
20          Dim standardIcon As IPictureDisp = PictureDispConverter.ToIPictureDisp(My.Resources.Cat_16)
21          Dim toolTipImage As IPictureDisp = Nothing
22
23          'this is the text the user sees on the button
24          Dim buttonLabel As String = "API " & vbLf & "Help"
25
26          'text that displays when the user hovers over the button
27          Dim toolTip_Simple As String = "Opens the API Help"
28          Dim toolTip_Expanded As String = Nothing
29
30
31      Progressive ToolTip
47
48          Dim buttonDef As ButtonDefinition
49          buttonDef = CreateButtonDefintion.CreateButtonDef(environment, CustomDrawingTab, ribbonPanel, useLargeIcon,
50                                                            isInButtonStack, useProgressToolTip,
51                                                            buttonLabel, toolTip_Simple, toolTip_Expanded,
52                                                            standardIcon, largeIcon, toolTipImage)
53
54          Return buttonDef
55
56      End Function


77
78  End Module
79
```

Specify the Button Display Name



Understanding the Add-in Template Structure

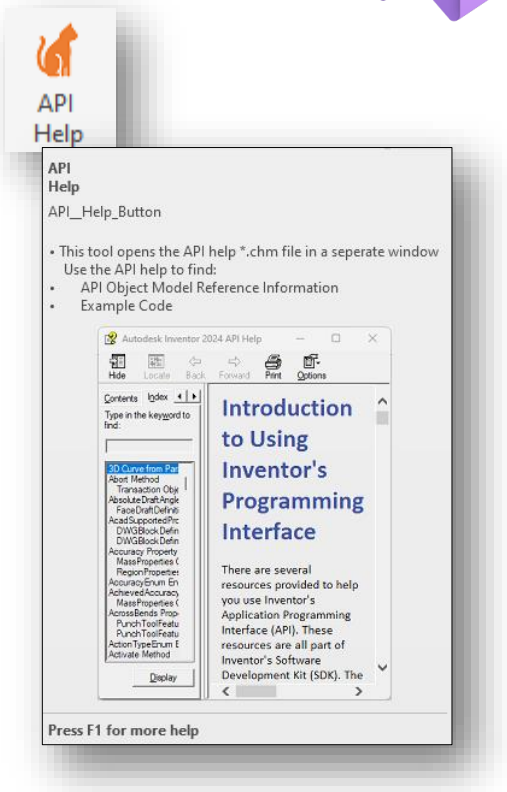
API_Help.vb*

My_New_Addin | API_Help | CreateButton

```
1   Imports Inventor
2
    4 references
3   Module API_Help
4
5       ''' <summary> Creates a button Button Definition Define the label, icons, tool t ...
13
14      'This function is where the button is defined
        3 references
15      Function CreateButton(environment As String, CustomDrawingTab As RibbonTab, ribbonPanel As RibbonPanel,
16                            useLargeIcon As Boolean, isInButtonStack As Boolean) As ButtonDefinition
17
18          'get the images to use for the button
19          Dim largeIcon As IPictureDisp = PictureDispConverter.ToIPictureDisp(My.Resources.Cat_32)
20          Dim standardIcon As IPictureDisp = PictureDispConverter.ToIPictureDisp(My.Resources.Cat_16)
21          Dim toolTipImage As IPictureDisp = Nothing
22
23          'this is the text the user sees on the button
24          Dim buttonLabel As String = "API " & vbLf & "Help"
25
26          'text that displays when the user hovers over the button
27          Dim toolTip_Simple As String = "Opens the API Help"
28          Dim toolTip_Expanded As String = Nothing
29
30
31      Progressive ToolTip
47
48          Dim buttonDef As ButtonDefinition
49          buttonDef = CreateButtonDefintion.CreateButtonDef(environment, CustomDrawingTab, ribbonPanel, useLargeIcon,
50                                                            isInButtonStack, useProgressToolTip,
51                                                            buttonLabel, toolTip_Simple, toolTip_Expanded,
52                                                            standardIcon, largeIcon, toolTipImage)
53
54          Return buttonDef
55
56      End Function


77
78  End Module
79
```

Define the Tool Tip



Understanding the Add-in Template Structure

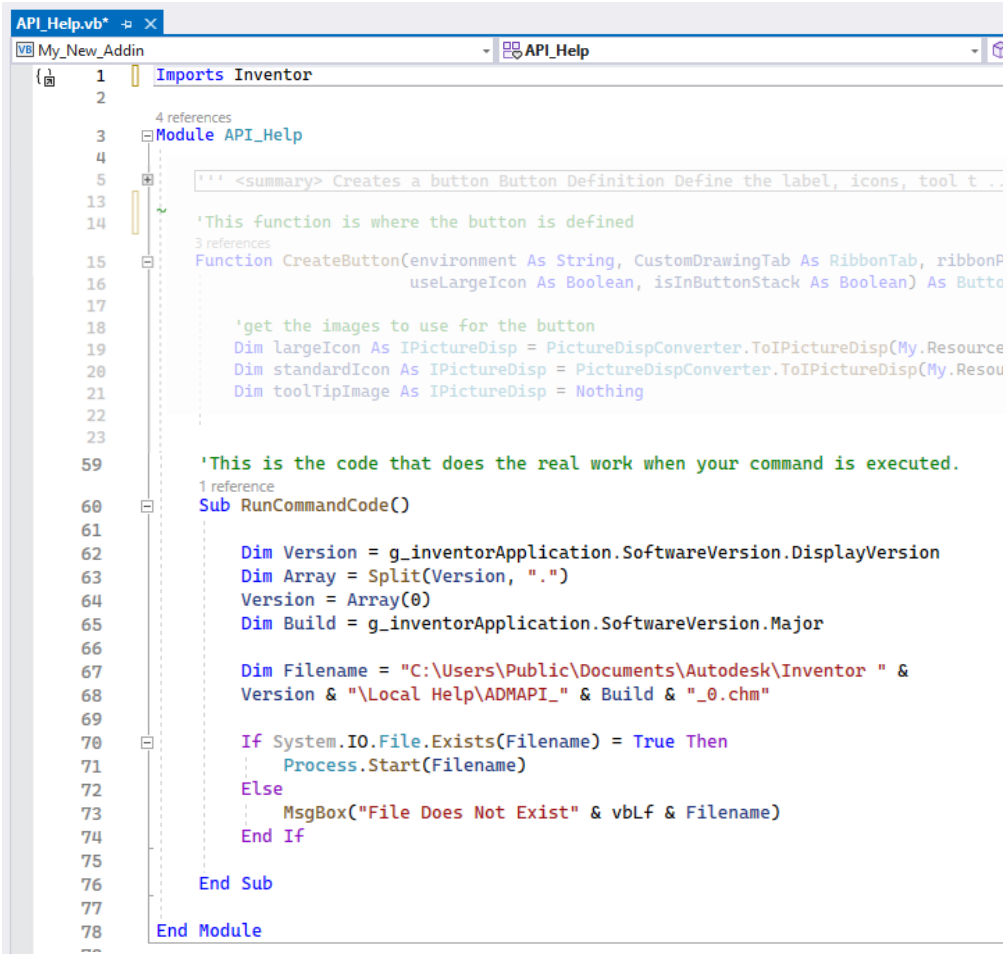
API_Help.vb* | My_New_Addin | API_Help | CreateButton

```
1   Imports Inventor
2
    4 references
3   Module API_Help
4
5       ''' <summary> Creates a button Button Definition Define the label, icons, tool t ...
13
14      'This function is where the button is defined
        3 references
15      Function CreateButton(environment As String, CustomDrawingTab As RibbonTab, ribbonPanel As RibbonPanel,
16                            useLargeIcon As Boolean, isInButtonStack As Boolean) As ButtonDefinition
17
18          'get the images to use for the button
19          Dim largeIcon As IPictureDisp = PictureDispConverter.ToIPictureDisp(My.Resources.Cat_32)
20          Dim standardIcon As IPictureDisp = PictureDispConverter.ToIPictureDisp(My.Resources.Cat_16)
21          Dim toolTipImage As IPictureDisp = Nothing
22
23          'this is the text the user sees on the button
24          Dim buttonLabel As String = "API " & vbLf & "Help"
25
26          'text that displays when the user hovers over the button
27          Dim toolTip_Simple As String = "Opens the API Help"
28          Dim toolTip_Expanded As String = Nothing
29
30
31      Progressive ToolTip
47
48          Dim buttonDef As ButtonDefinition
49          buttonDef = CreateButtonDefintion.CreateButtonDef(environment, CustomDrawingTab, ribbonPanel, useLargeIcon,
50                                                            isInButtonStack, useProgressToolTip,
51                                                            buttonLabel, toolTip_Simple, toolTip_Expanded,
52                                                            standardIcon, largeIcon, toolTipImage)
53
54          Return buttonDef
55
56      End Function


77
78  End Module
79
```

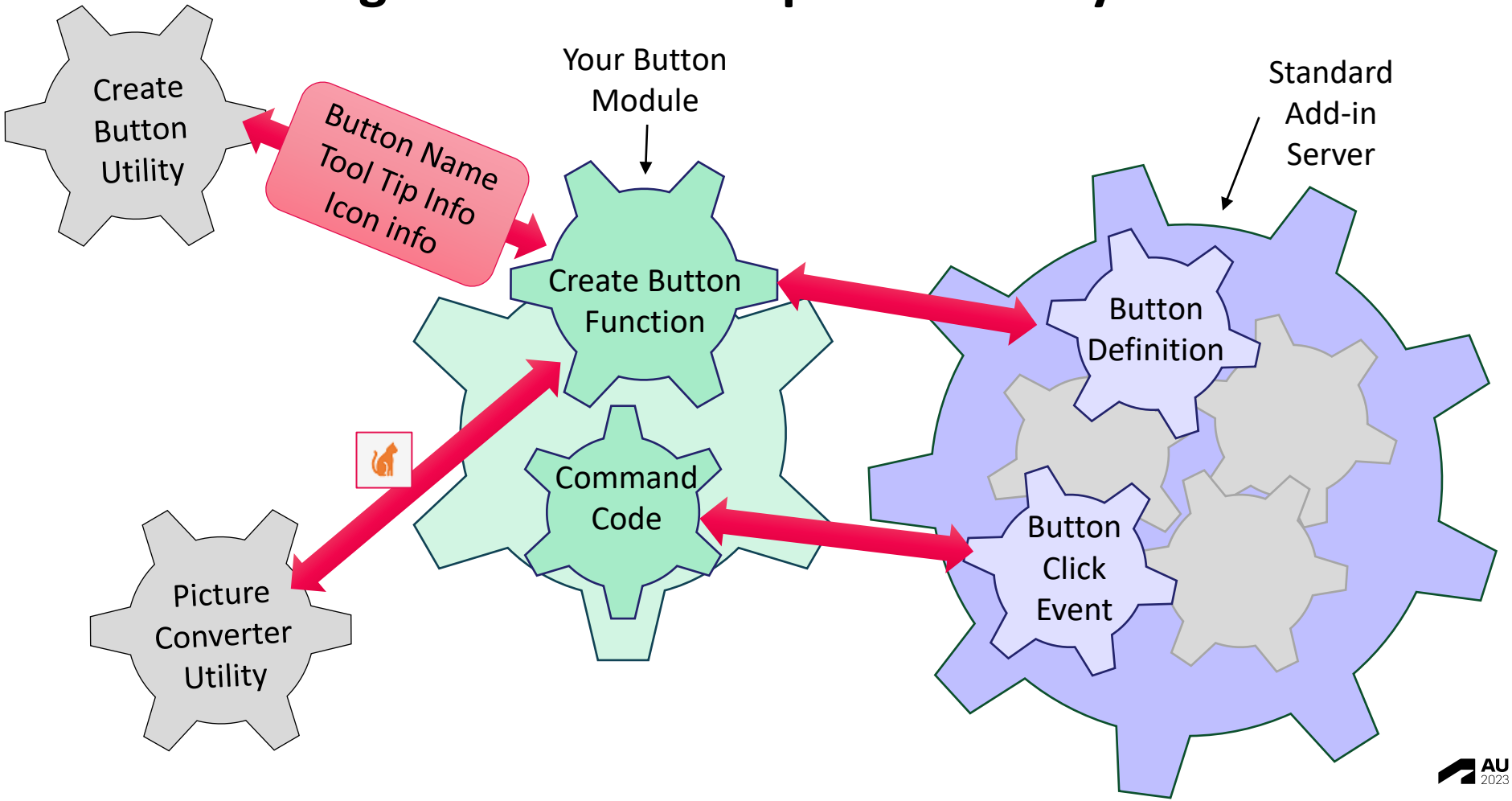
Pass the information to the utility module

Understanding the Add-in Template Structure

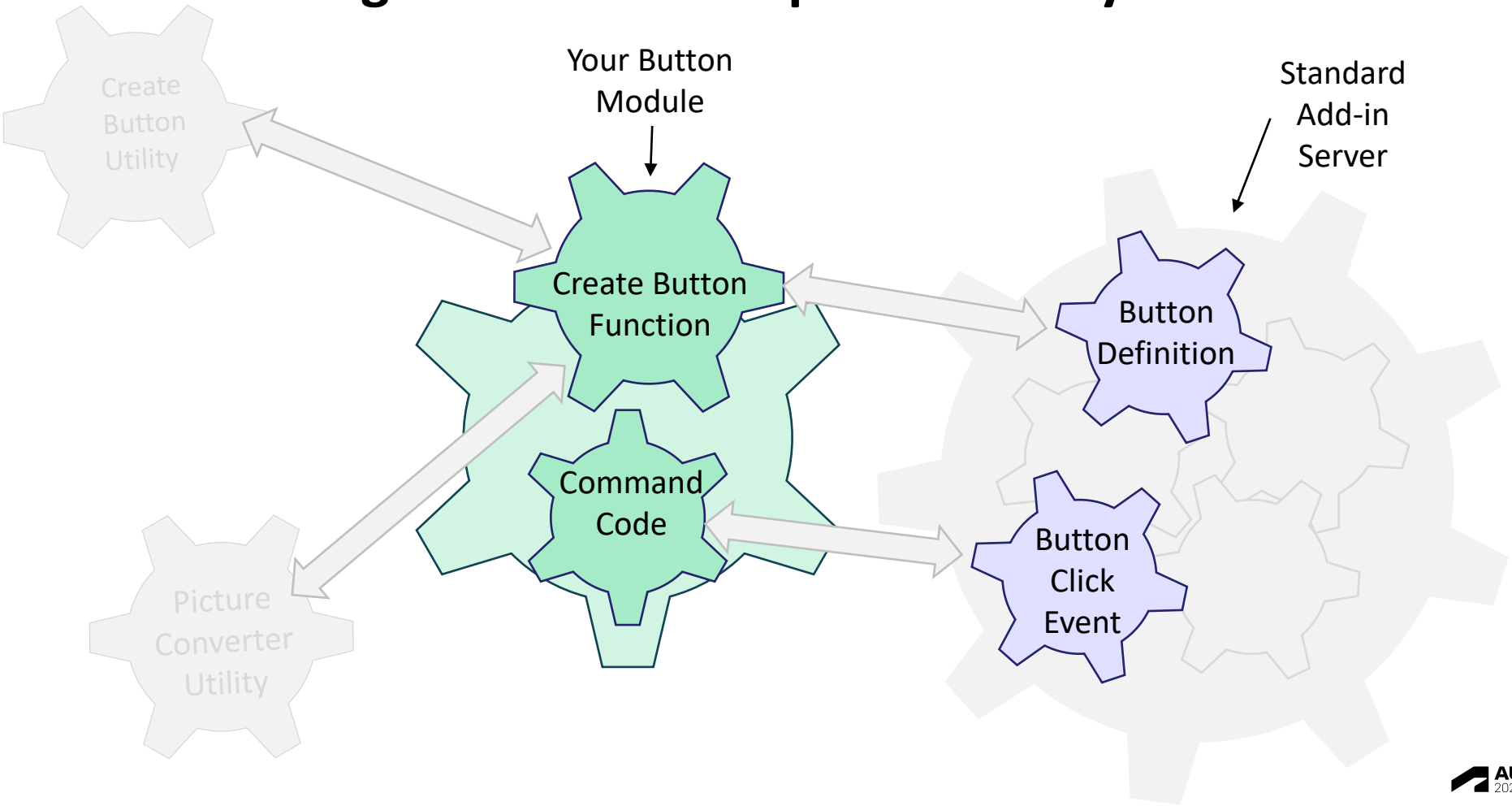


- One module for each button.
  - Function to define the button
  - Sub to house the code that is run when the button is clicked

Understanding the Add-in Template Visually

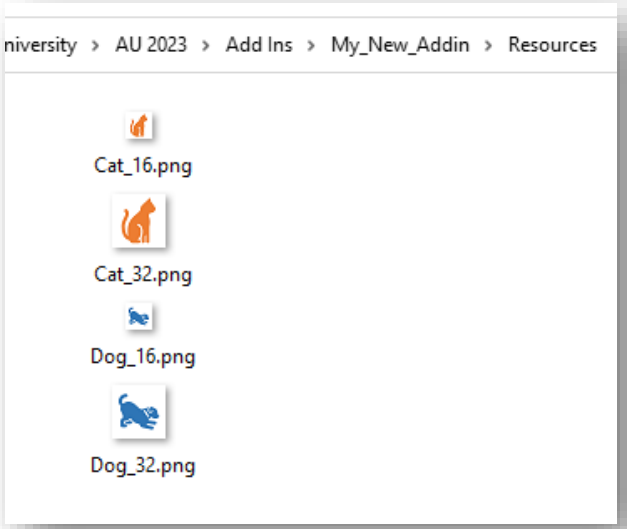


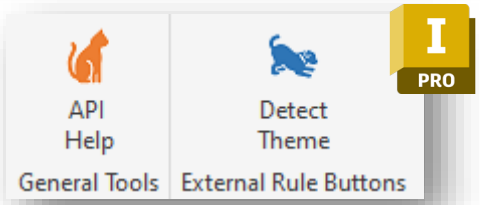
Understanding the Add-in Template Visually



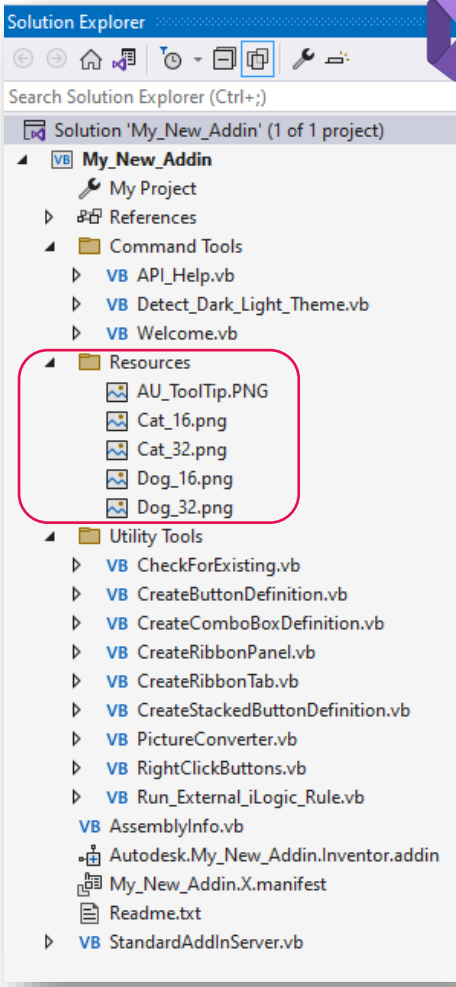
Understanding the Add-in Template Structure

- A folder that holds the pictures used for the button icons





Resources →



Adding a button image

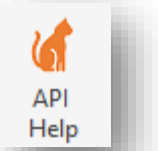
API_Help.vb*

My_New_Addin | API_Help | CreateButton

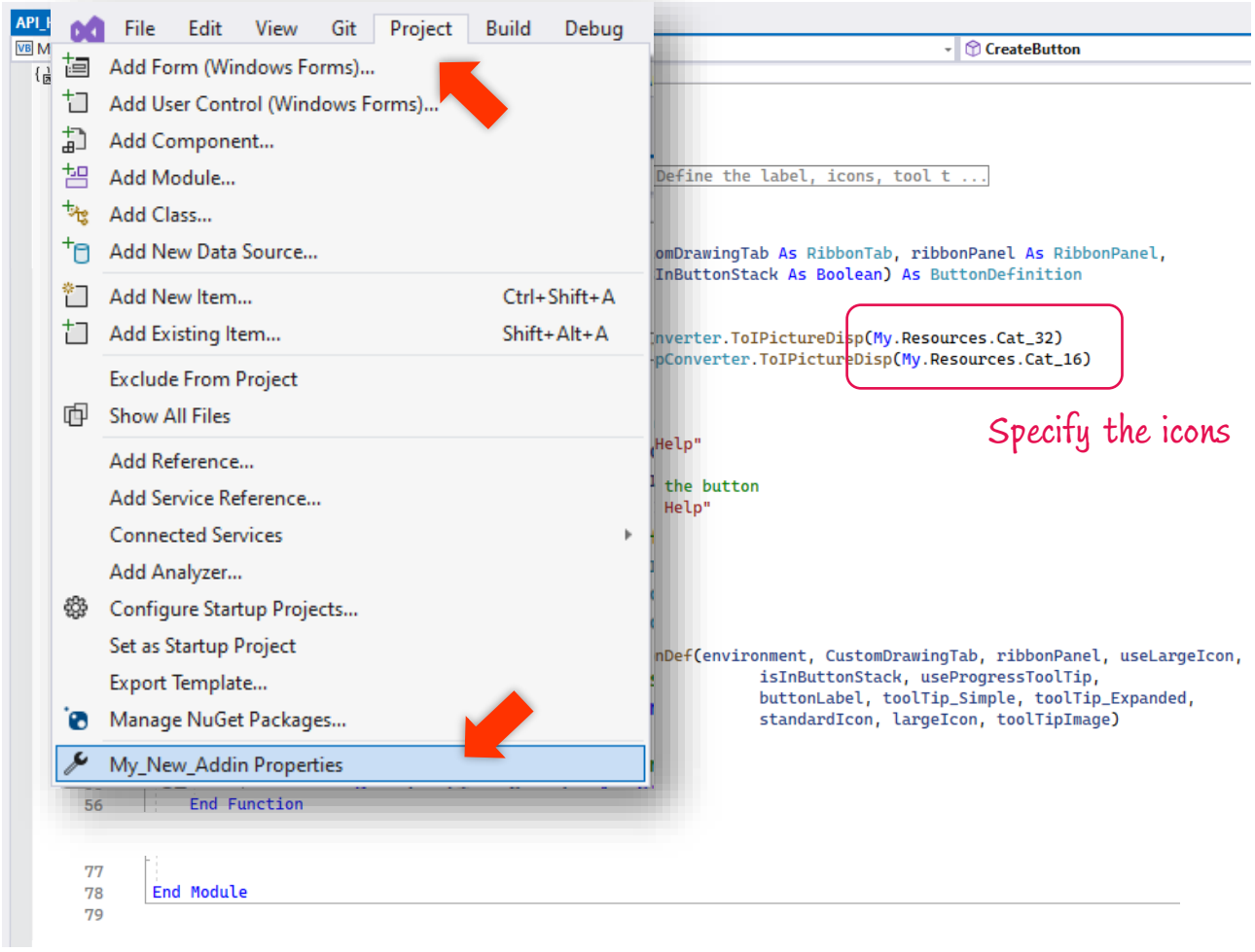
```
1   Imports Inventor
2
    4 references
3   Module API_Help
4
5       ''' <summary> Creates a button Button Definition Define the label, icons, tool t ...
13
14      'This function is where the button is defined
        3 references
15      Function CreateButton(environment As String, CustomDrawingTab As RibbonTab, ribbonPanel As RibbonPanel,
16                            useLargeIcon As Boolean, isInButtonStack As Boolean) As ButtonDefinition
17
18          'get the images to use for the button
19          Dim largeIcon As IPictureDisp = PictureDispConverter.ToIPictureDisp(My.Resources.Cat_32)
20          Dim standardIcon As IPictureDisp = PictureDispConverter.ToIPictureDisp(My.Resources.Cat_16)
21          Dim toolTipImage As IPictureDisp = Nothing
22
23          'this is the text the user sees on the button
24          Dim buttonLabel As String = "API " & vbLf & "Help"
25
26          'text that displays when the user hovers over the button
27          Dim toolTip_Simple As String = "Opens the API Help"
28          Dim toolTip_Expanded As String = Nothing
29
30
31      Progressive ToolTip
47
48          Dim buttonDef As ButtonDefinition
49          buttonDef = CreateButtonDefintion.CreateButtonDef(environment, CustomDrawingTab, ribbonPanel, useLargeIcon,
50                                                            isInButtonStack, useProgressToolTip,
51                                                            buttonLabel, toolTip_Simple, toolTip_Expanded,
52                                                            standardIcon, largeIcon, toolTipImage)
53
54          Return buttonDef
55
56      End Function


77
78  End Module
79
```

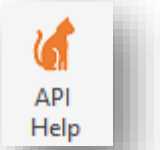
Specify the icons



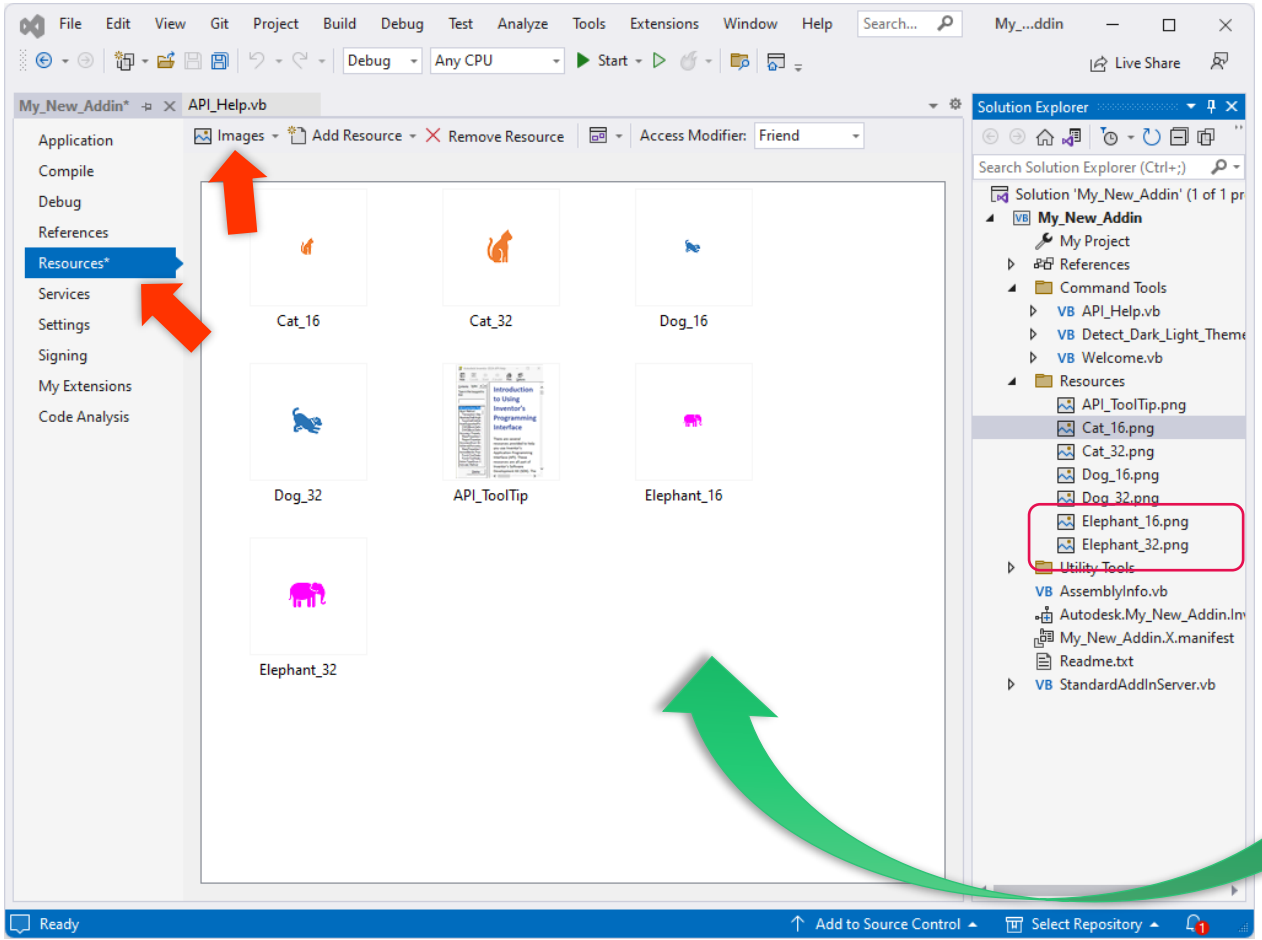
Adding a button image



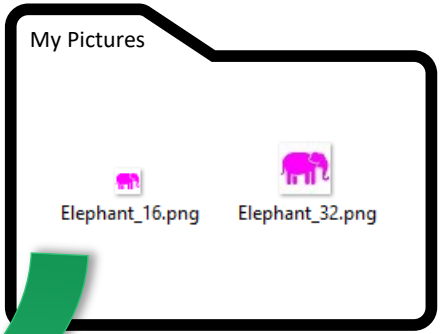
Specify the icons



Understanding the Add-in Template Structure



To add images, we simply drag and drop the images on the Resources pane



Understanding the Add-in Template Structure

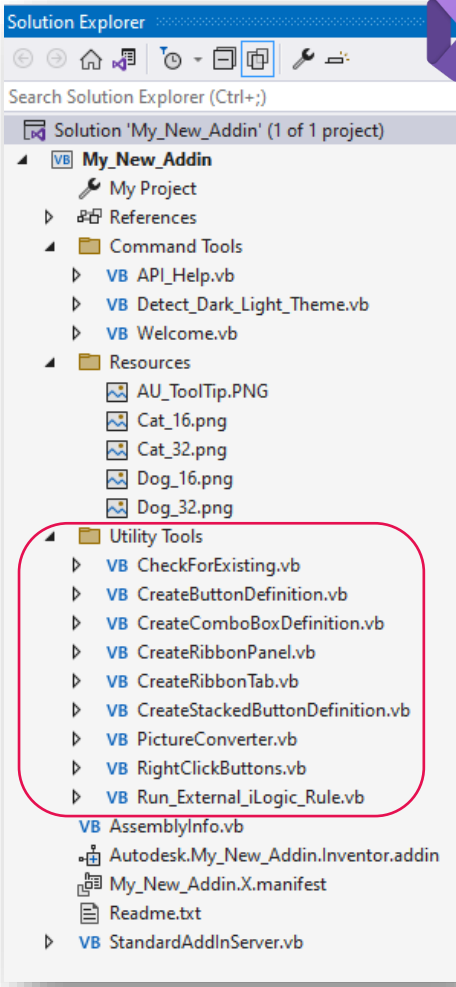
- Modules that are called over and over from the Command Tools and the StandardAddInServer

```
Dim buttonDef As ButtonDefinition
buttonDef = CreateButtonDefintion.CreateButtonDef(environment, CustomDrawingTab, ribbonPanel, useLargeIcon,
                                                  isInButtonStack, useProgressToolTip,
                                                  buttonLabel, toolTip_Simple, toolTip_Expanded,
                                                  standardIcon, largeIcon, toolTipImage)
```

```
'get the images to use for the button
Dim largeIcon As IPictureDisp = PictureDispConverter.ToIPictureDisp(My.Resources.Cat_32)
Dim standardIcon As IPictureDisp = PictureDispConverter.ToIPictureDisp(My.Resources.Cat_16)
```

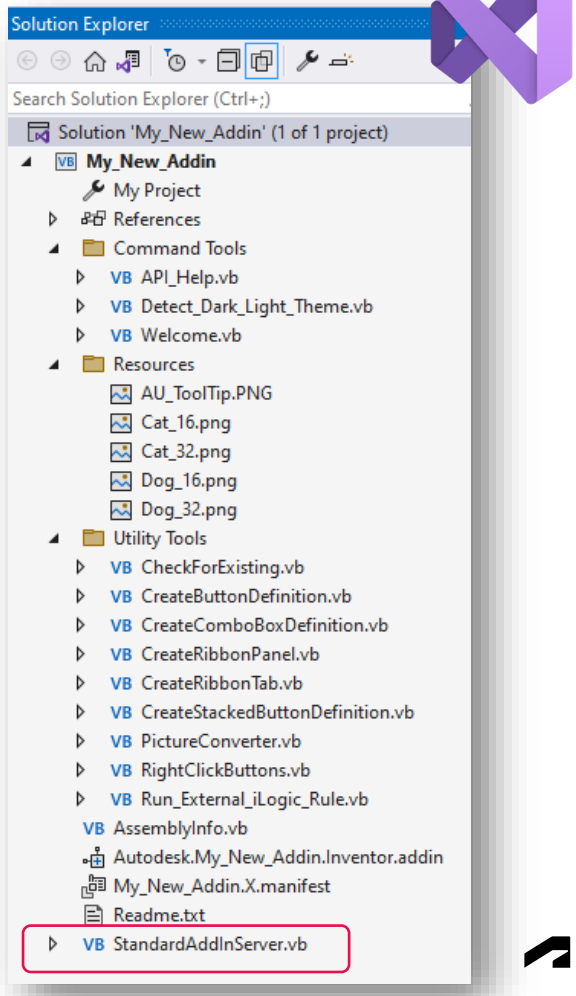
```
Run_External_iLogic_Rule.RunExternalRule("Detect Dark or Light Theme")
```

Utility Modules →



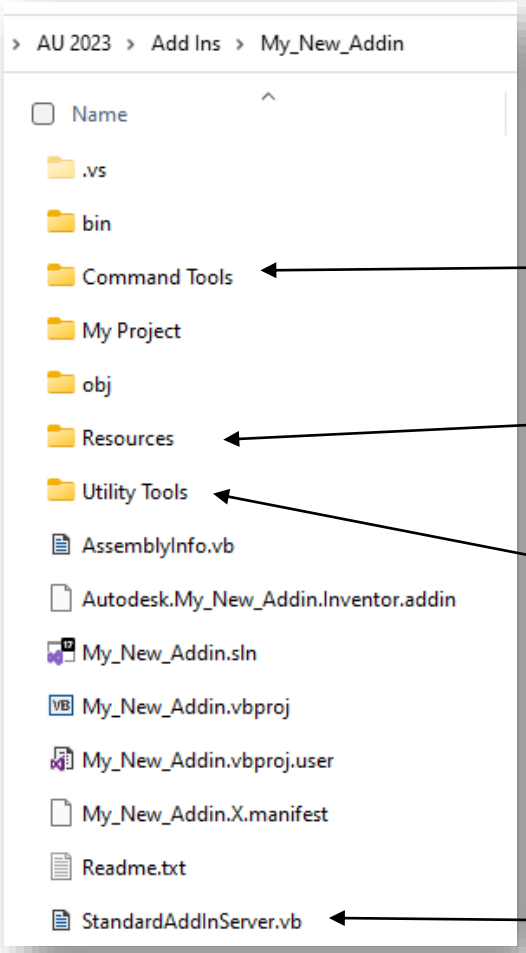
Understanding the Add-in Template Structure

- The primary file that defines the add-in
- This is what Inventor calls when it loads the add-in
- This is what connects our command modules to Inventor
- This is what handles the events:
  - Document OnOpen
  - Document OnSave
  - Button OnExecute
  - … and so on.



Standard Add-in Server →

Understanding the Add-in Template Structure

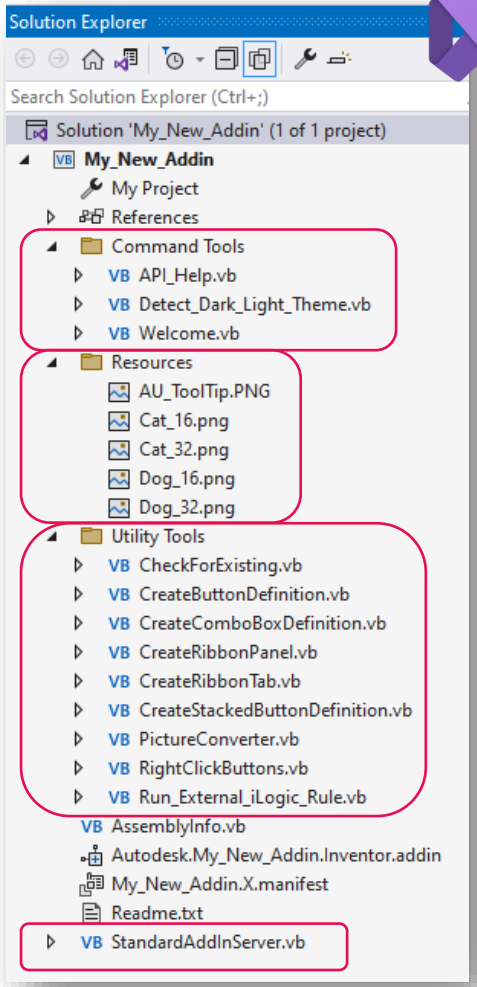


Command Tool Modules

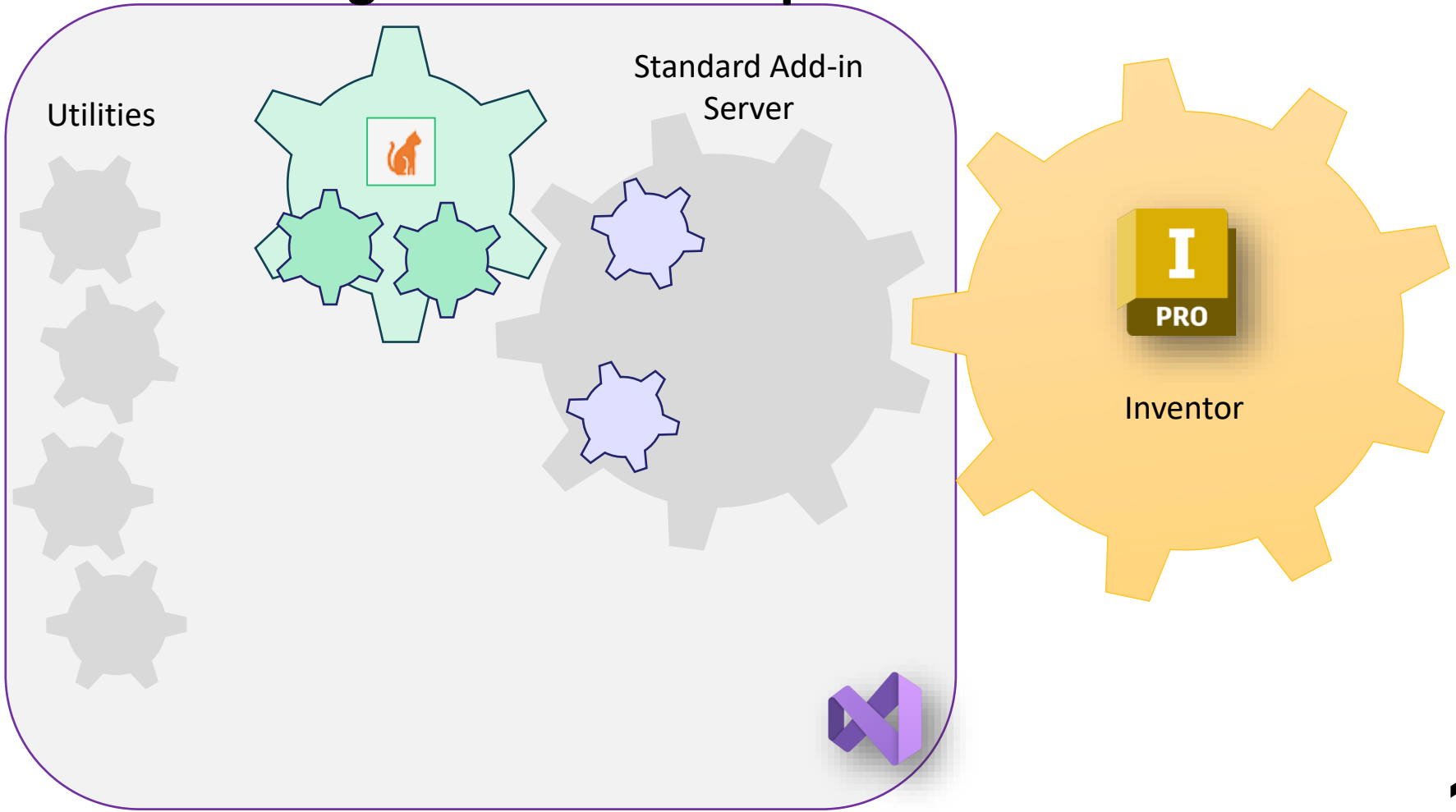
Resources

Utility Modules

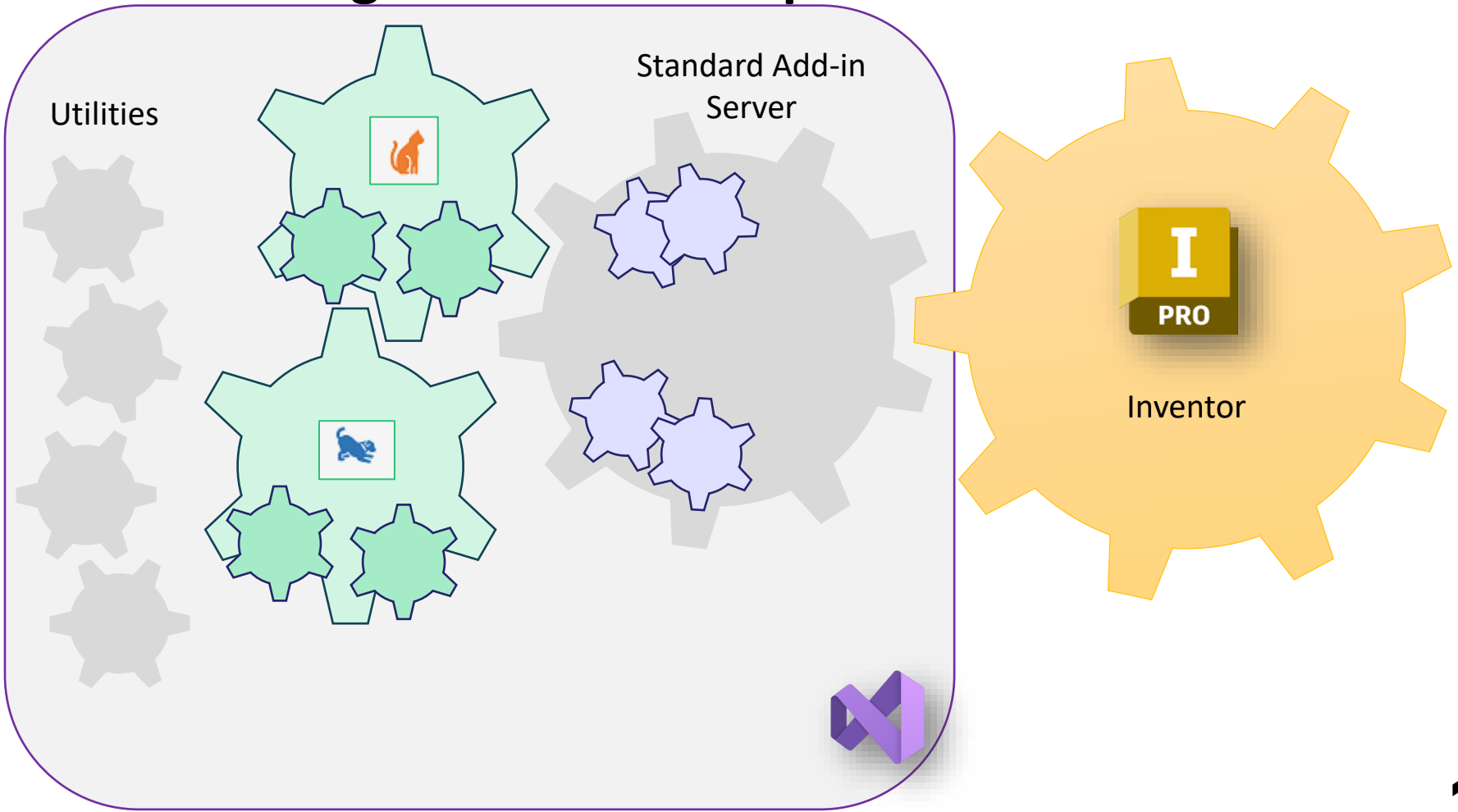
Standard Add-in Server



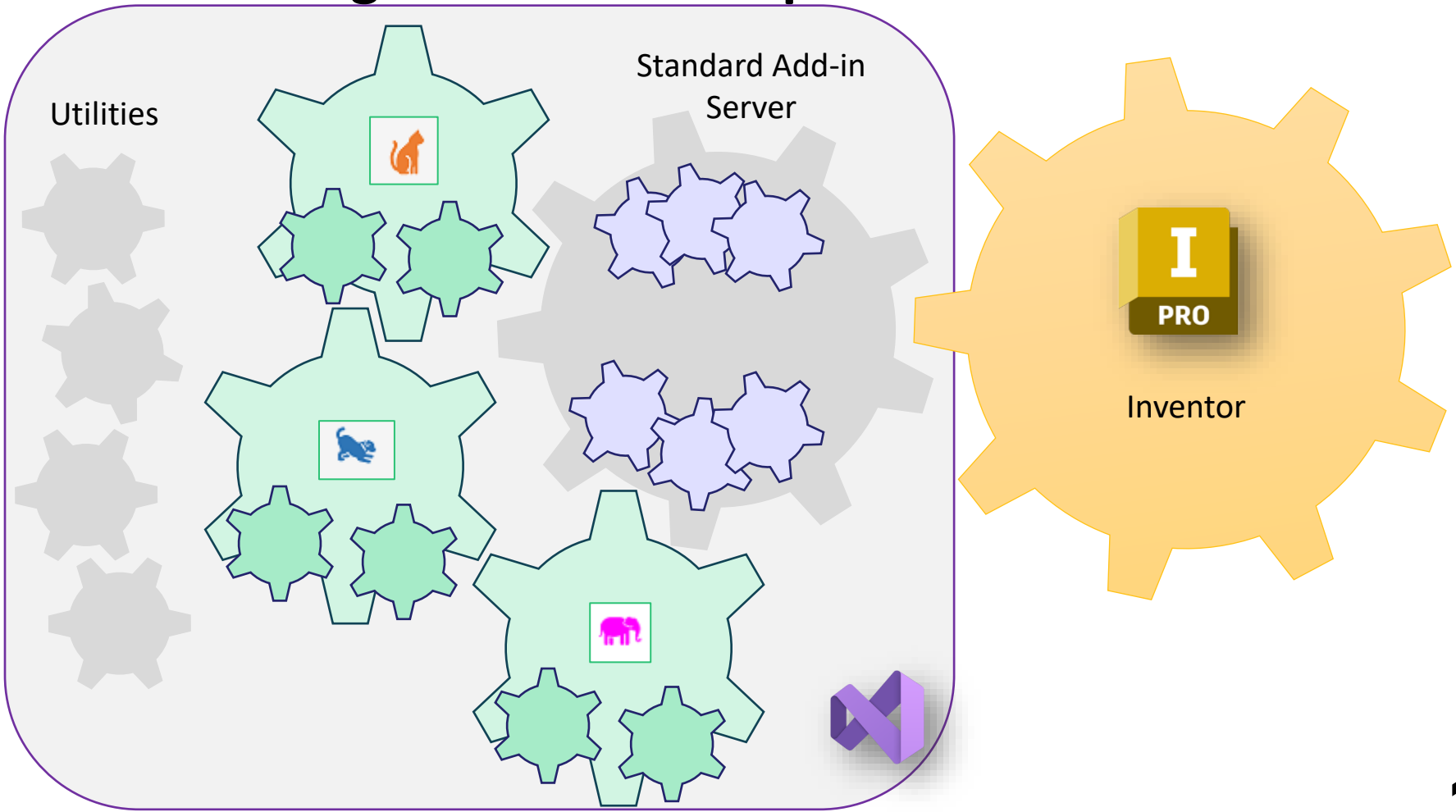
Understanding the Add-in Template Structure



Understanding the Add-in Template Structure



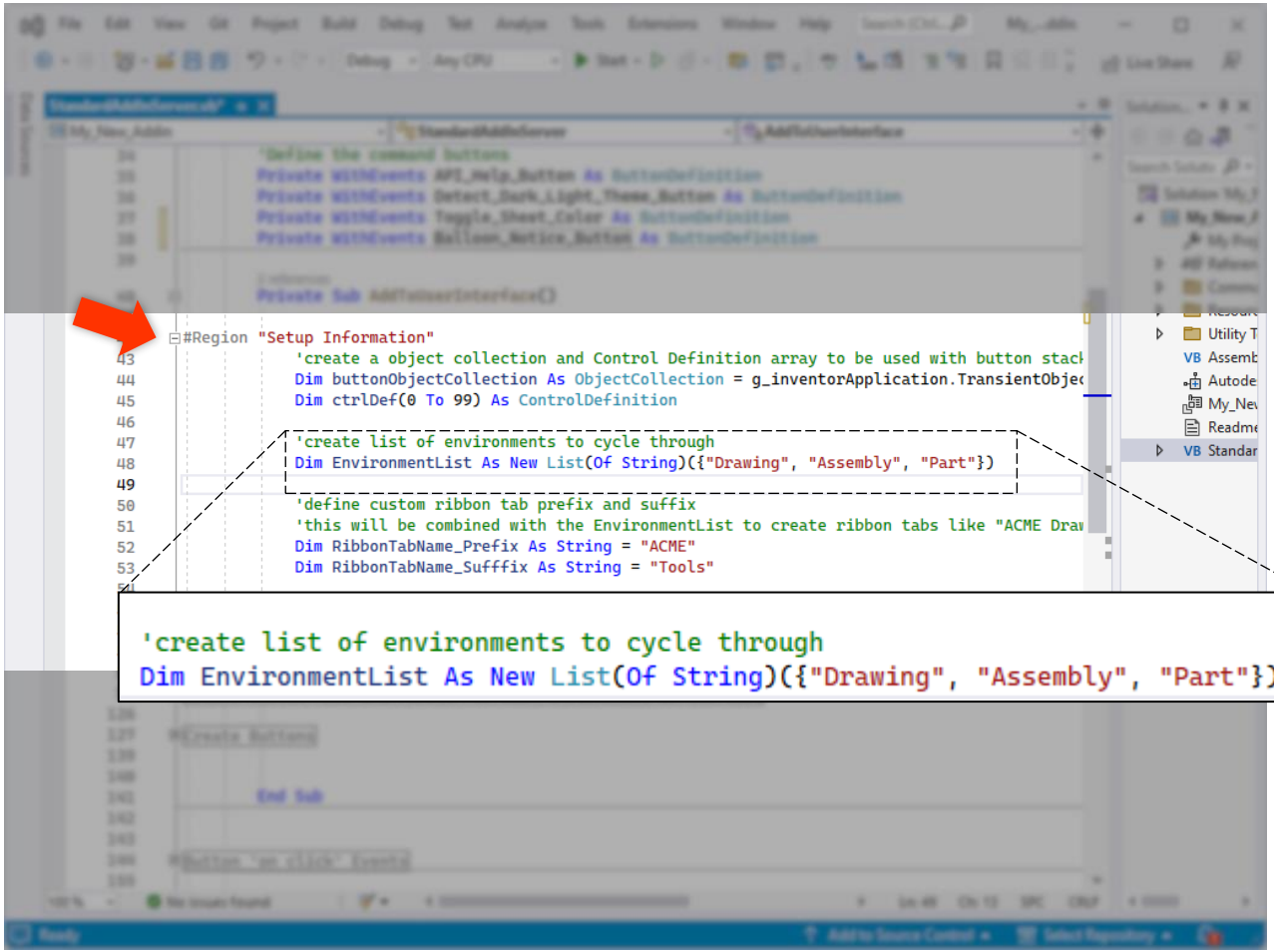
Understanding the Add-in Template Structure



Modifying the Add-in

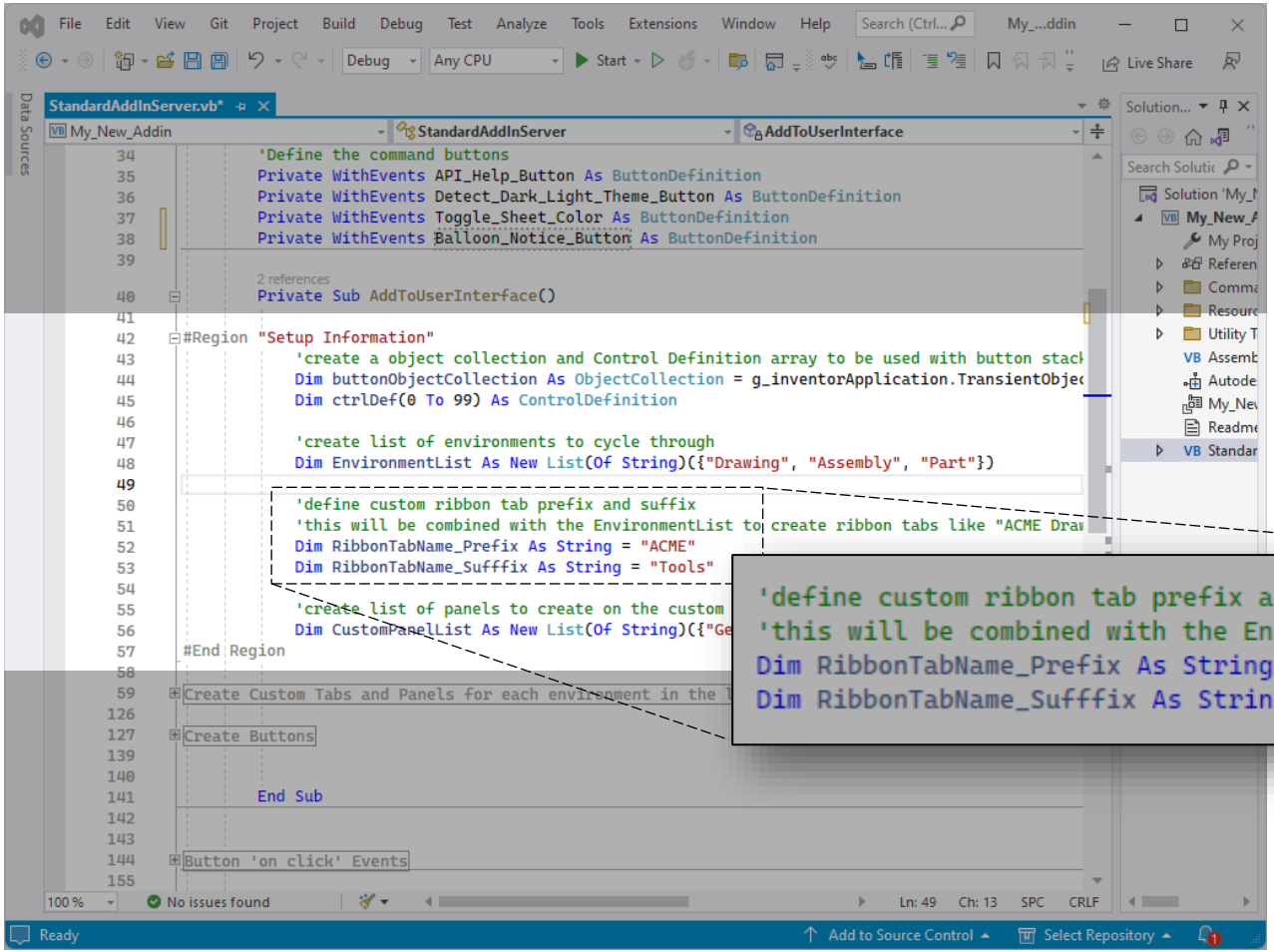
Working in Visual Studio

Setup Information



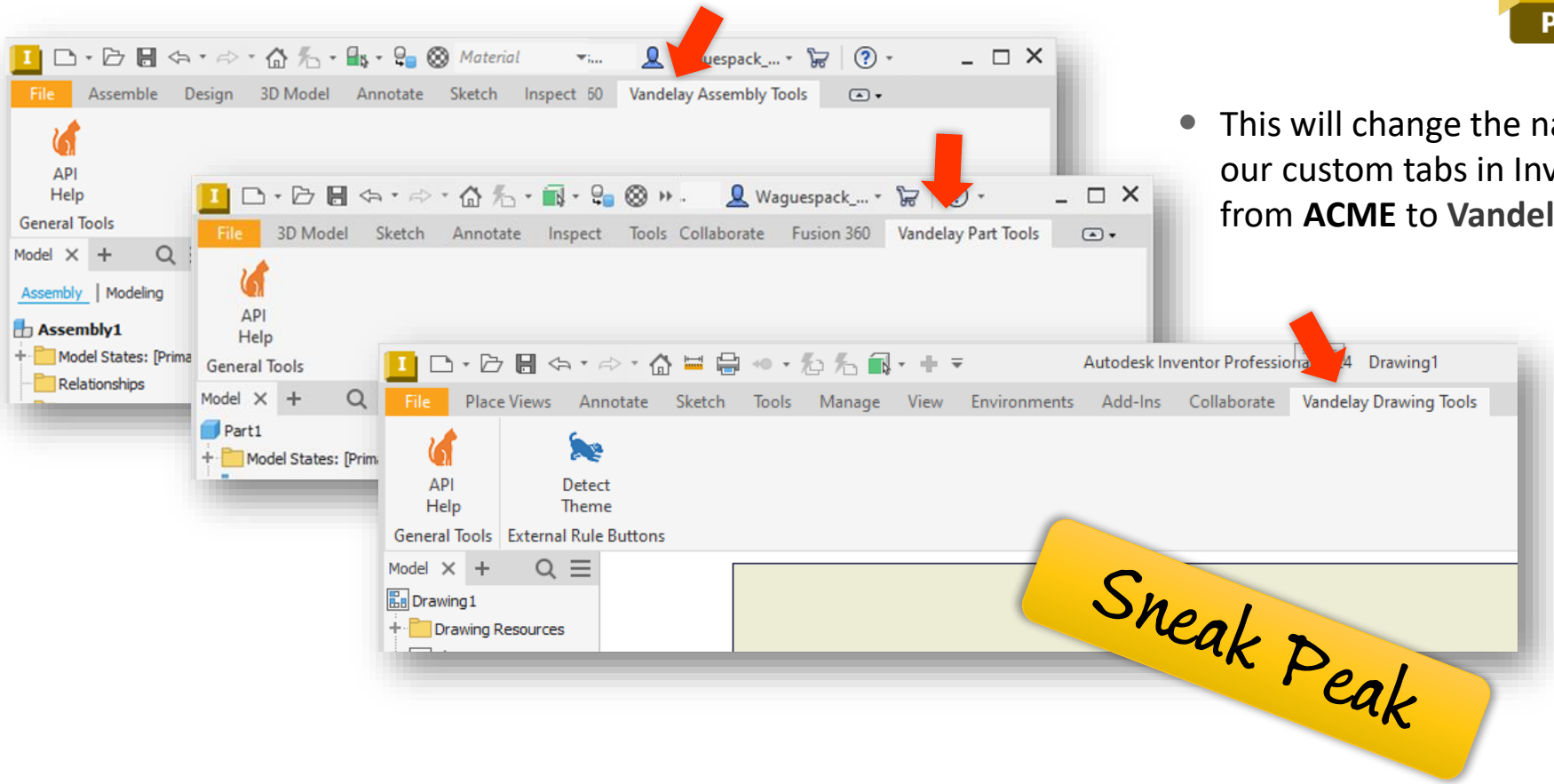
- We double click the **StandardAddinServer** file in the Solution Explorere to open it for edits.
- If we expand the **Setup Information** region, we can see where the custom tabs come from
- If we wanted the custom tab in only the Drawing and Part environments, we could modify the EnvironmentList and remove Assembly

Setup Information



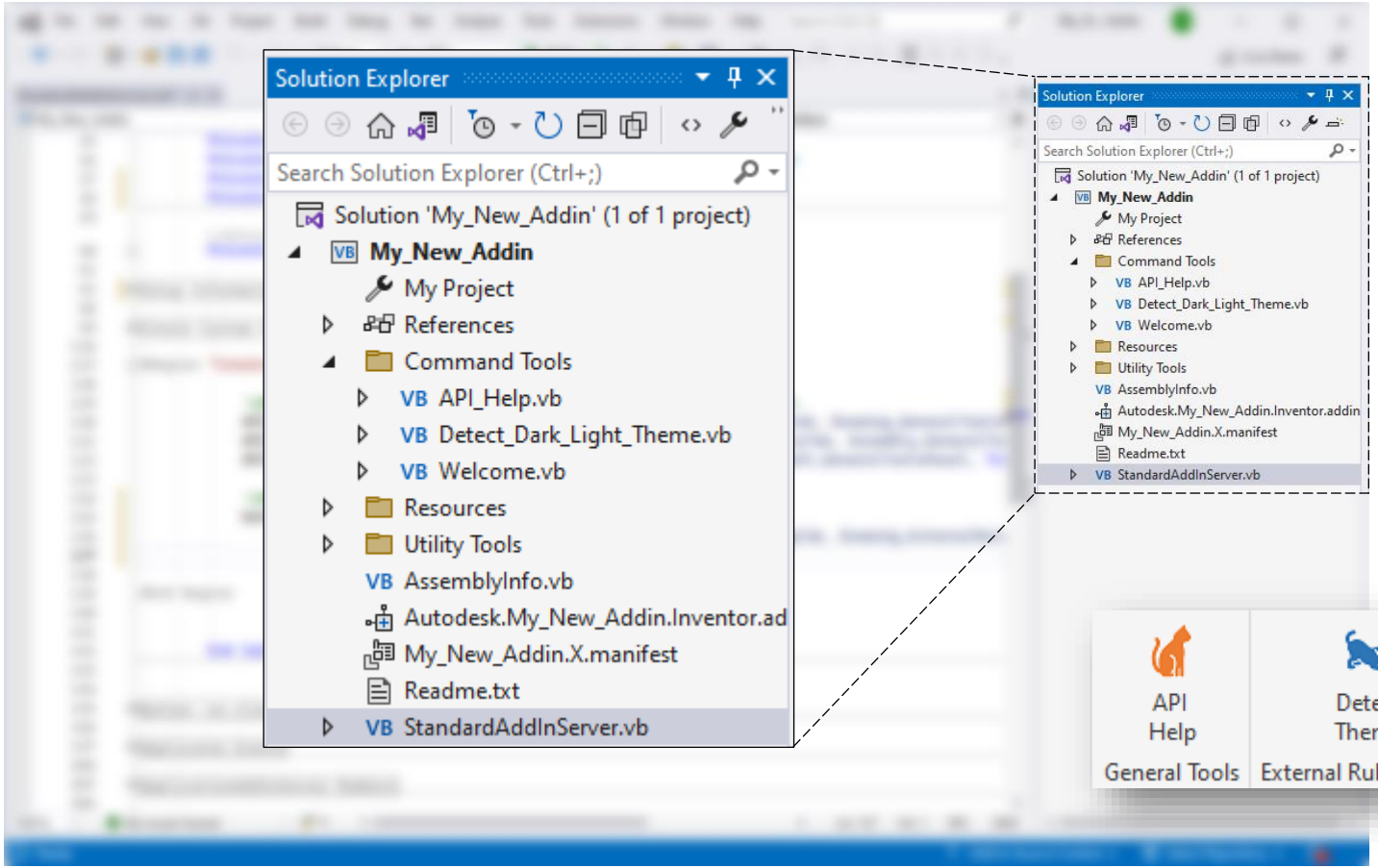
- We can see where the name of the tab is set here.
- We'll change that from **ACME** to **Vandelay**

Renamed Custom Tab



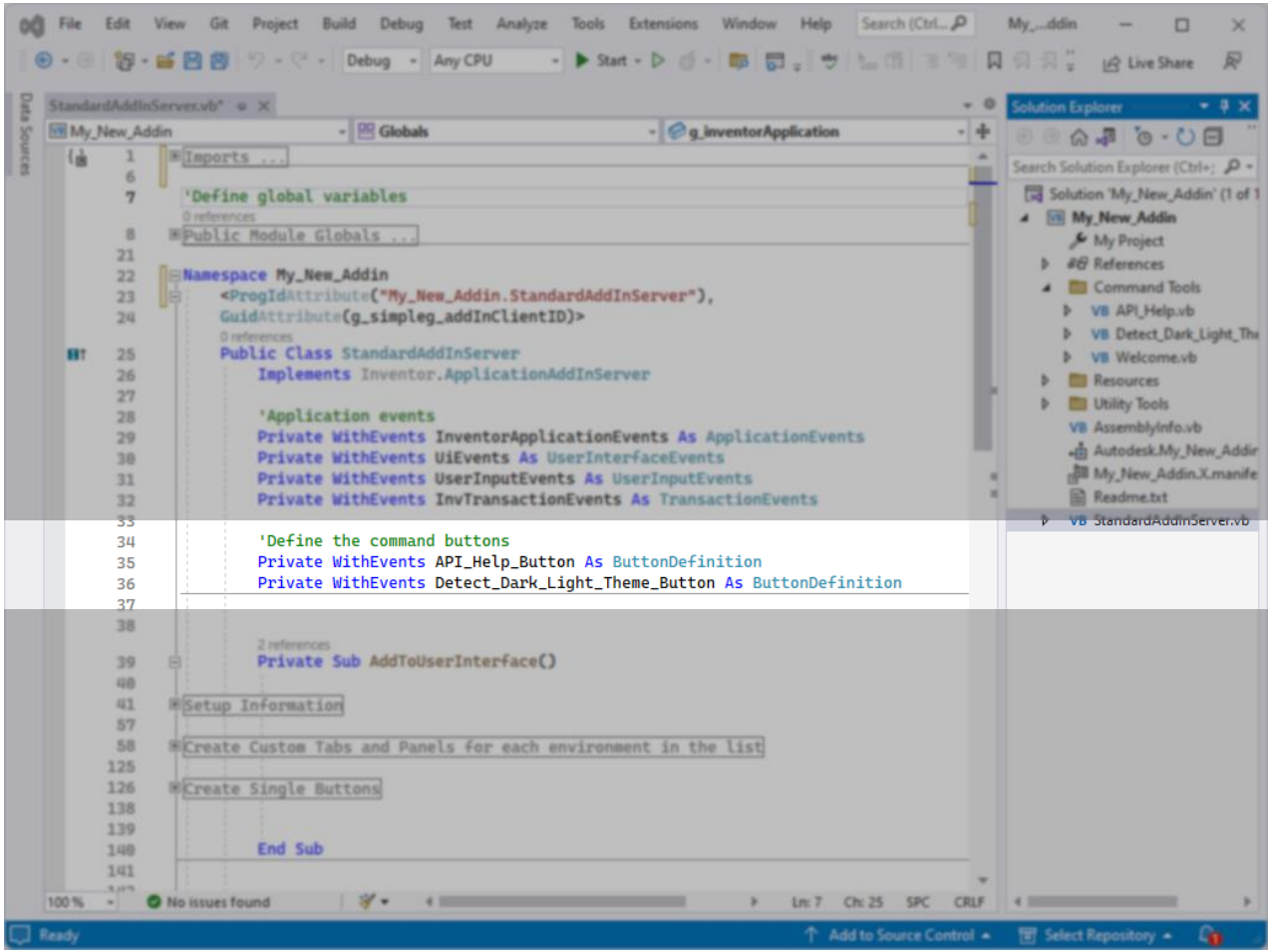
- This will change the name of our custom tabs in Inventor from **ACME** to **Vandelay**

Modifying the Add-in



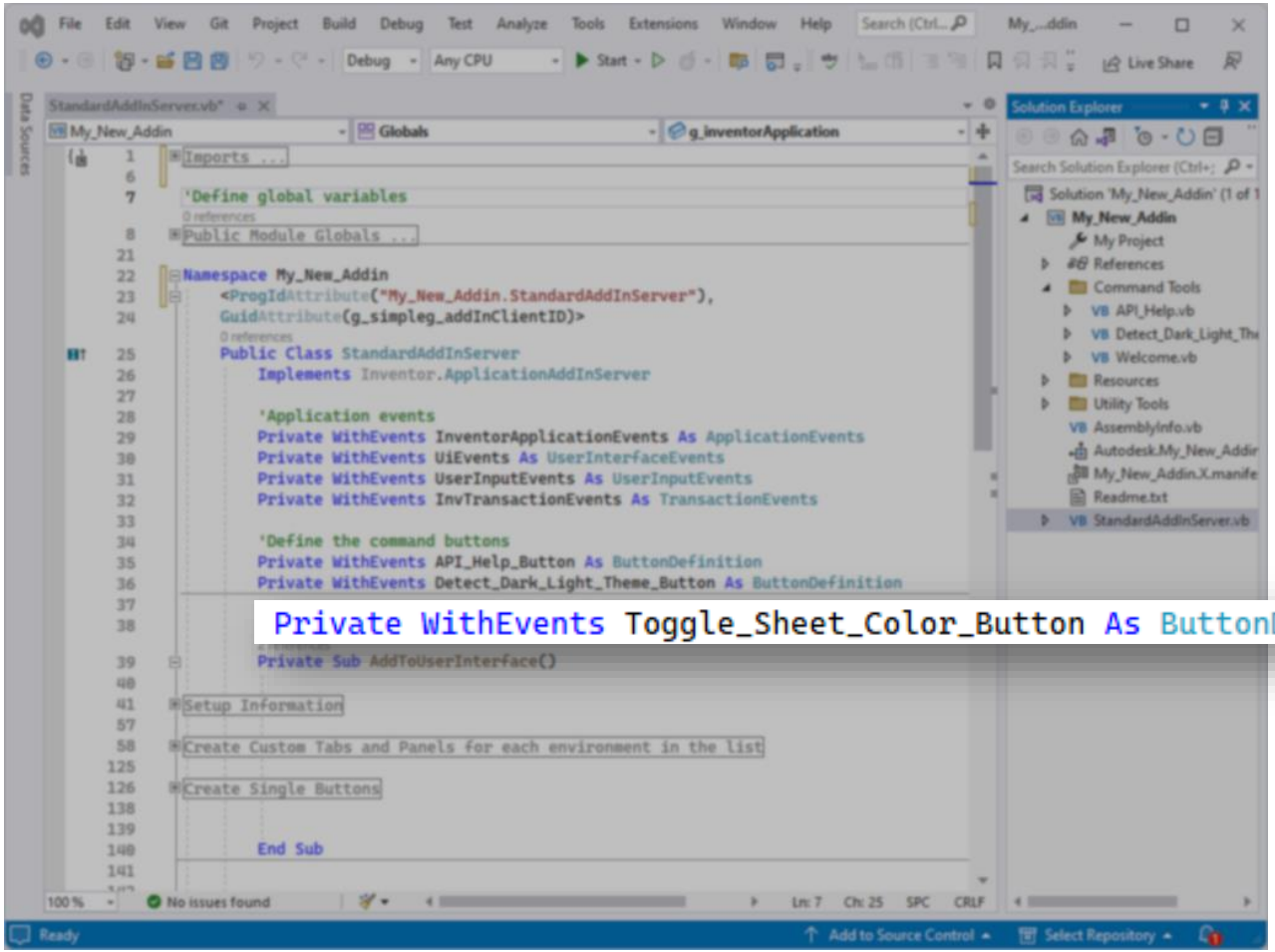
- If we expand the **Create Buttons** region, we can see where the buttons are created

Modifying the Add-in



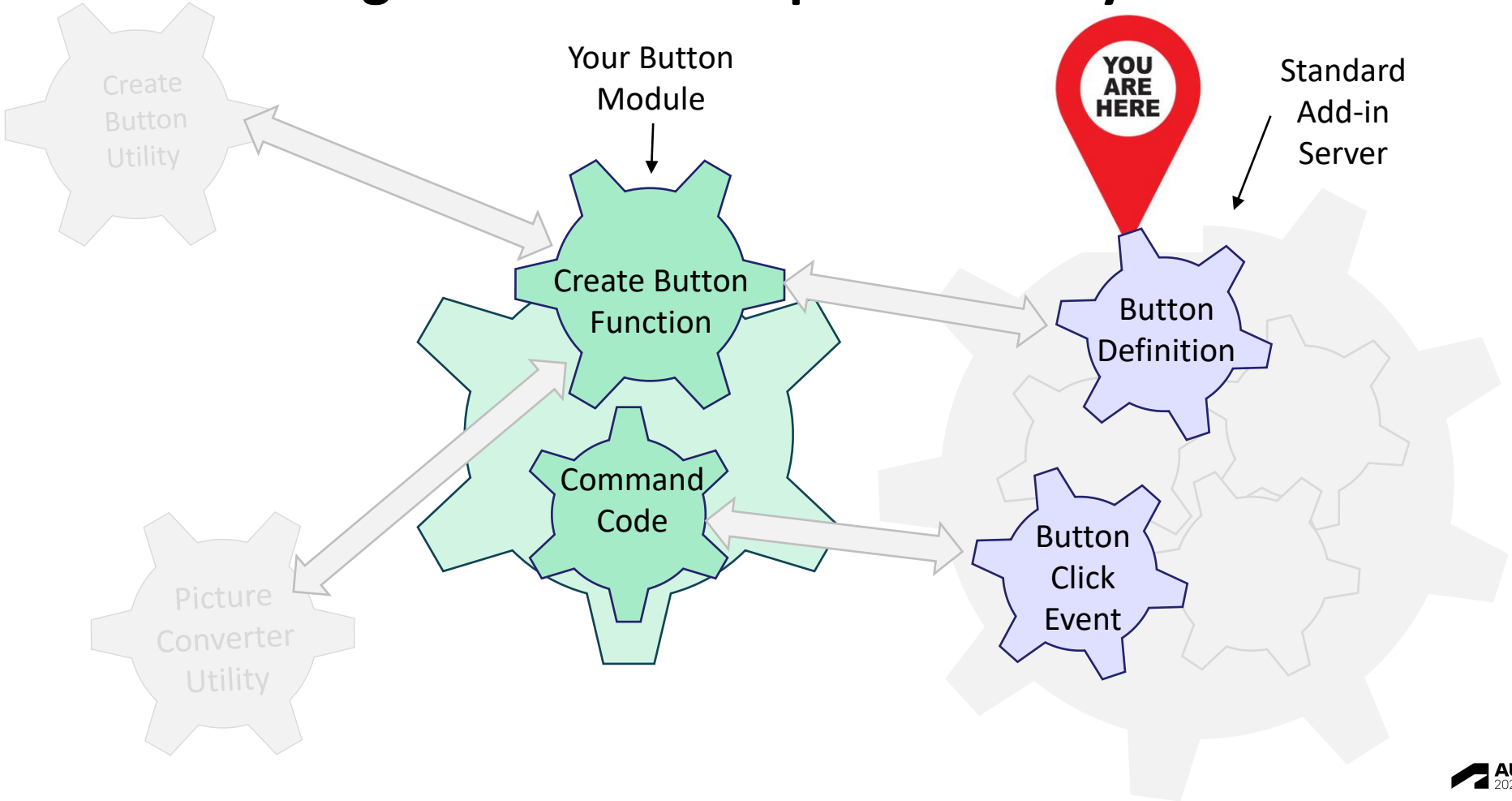
- Still in the **StandardAddinServer** file…
- Note the area at the top where the buttons are defined.
- We'll add a new button

Modifying the Add-in

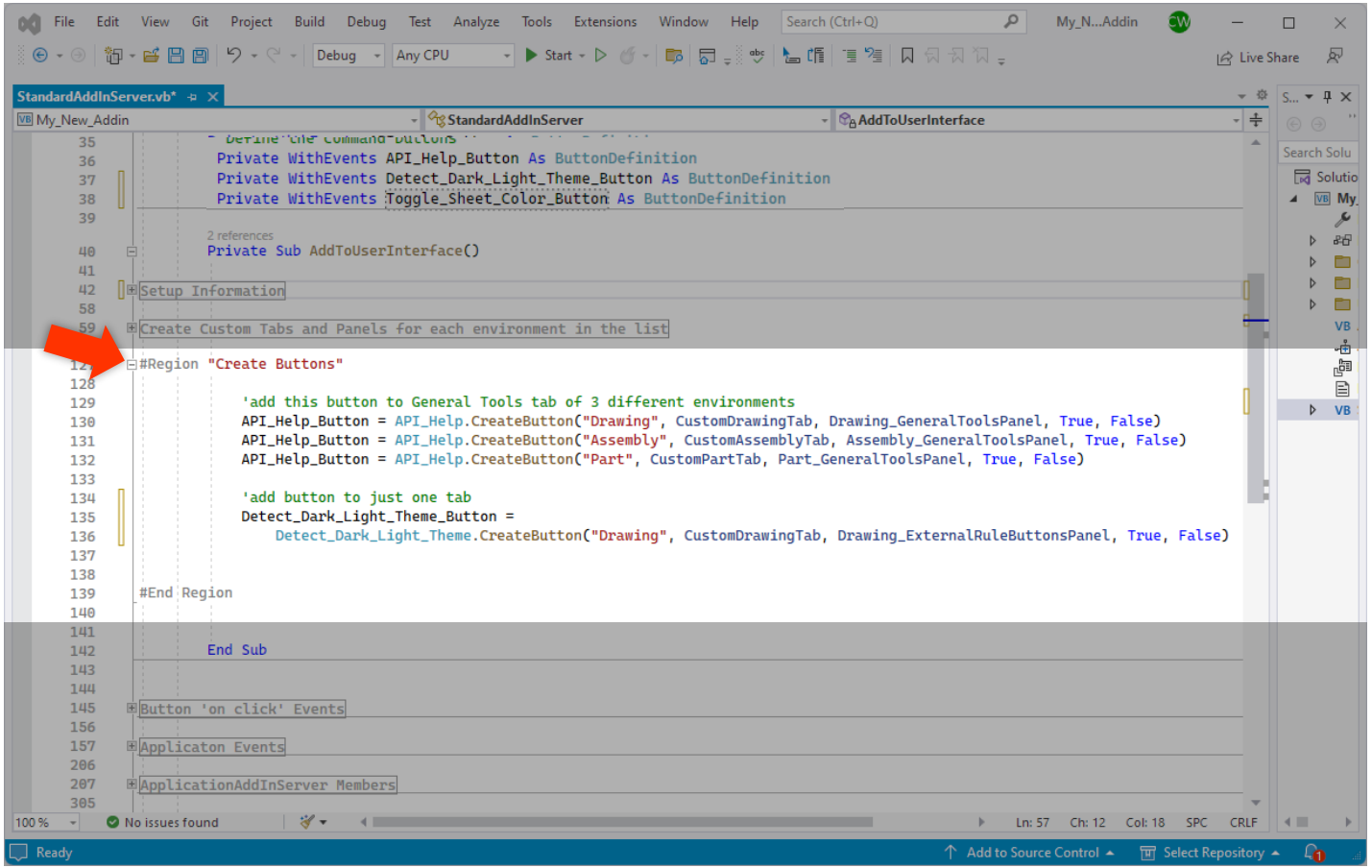


- We'll add a new button called:
  - Toggle_Sheet_Color_Button

Understanding the Add-in Template Visually

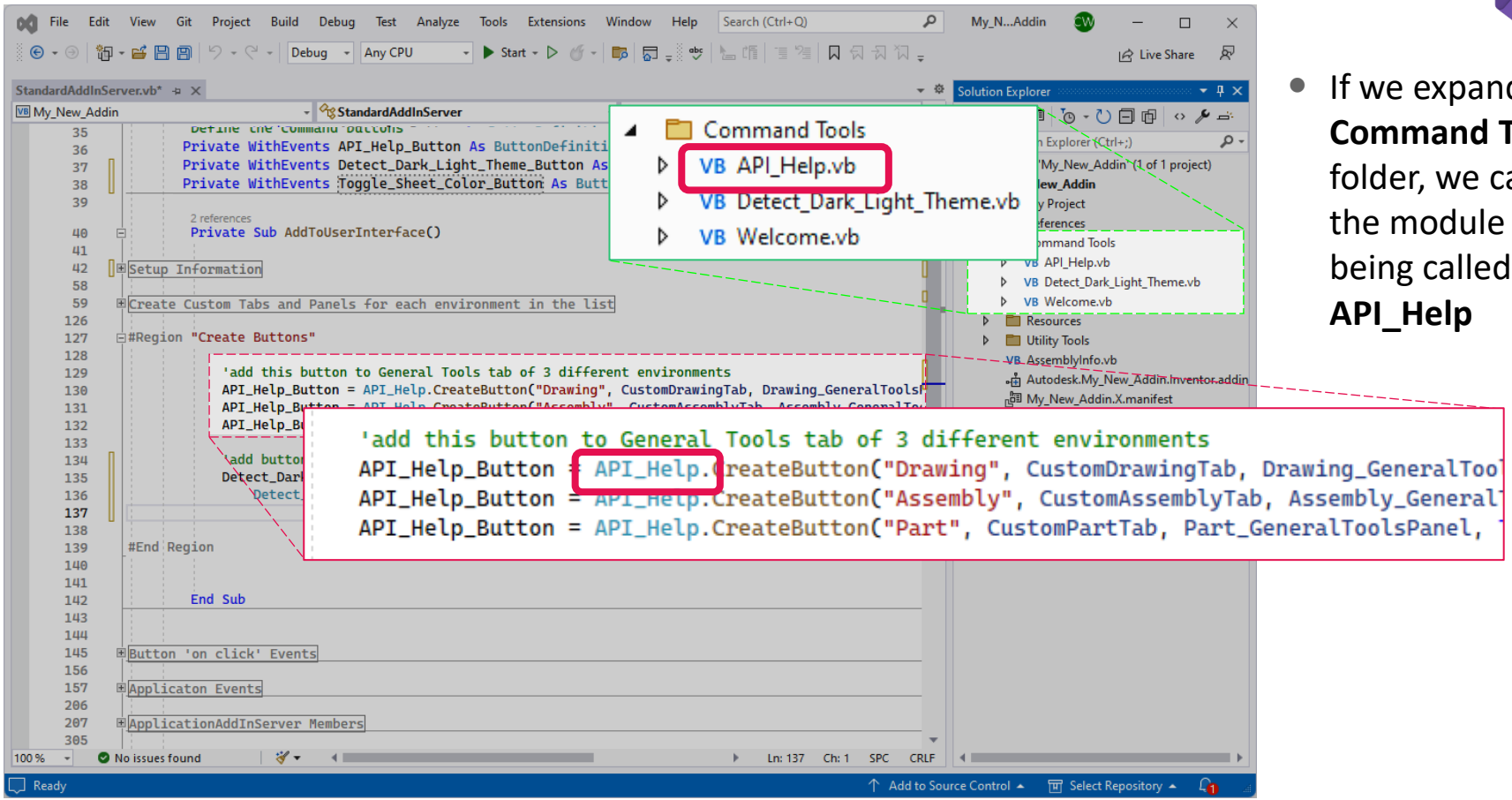


Modifying the Add-in



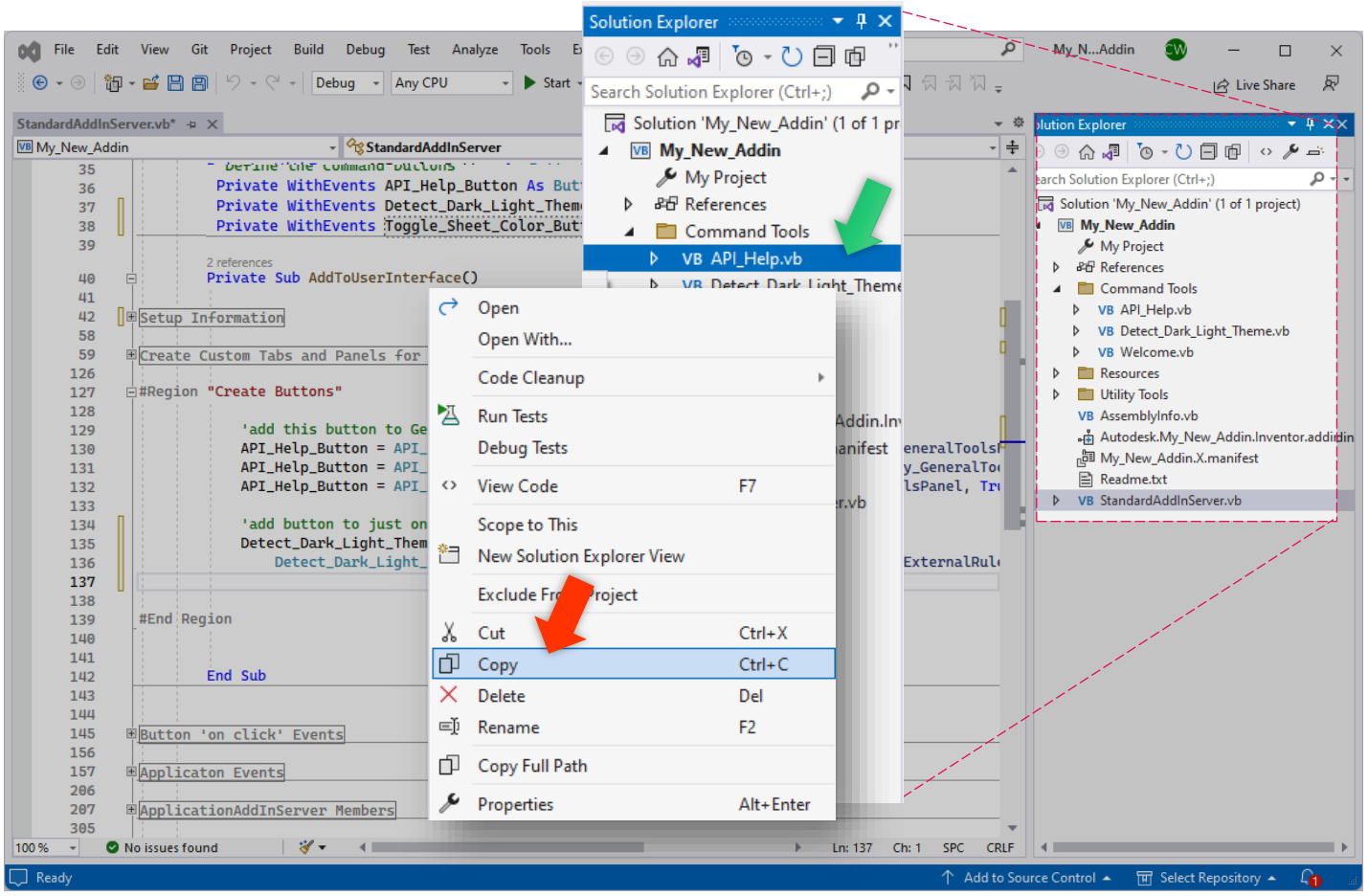
- If we expand the **Create Buttons** region, we can see where the buttons are created

Modifying the Add-in



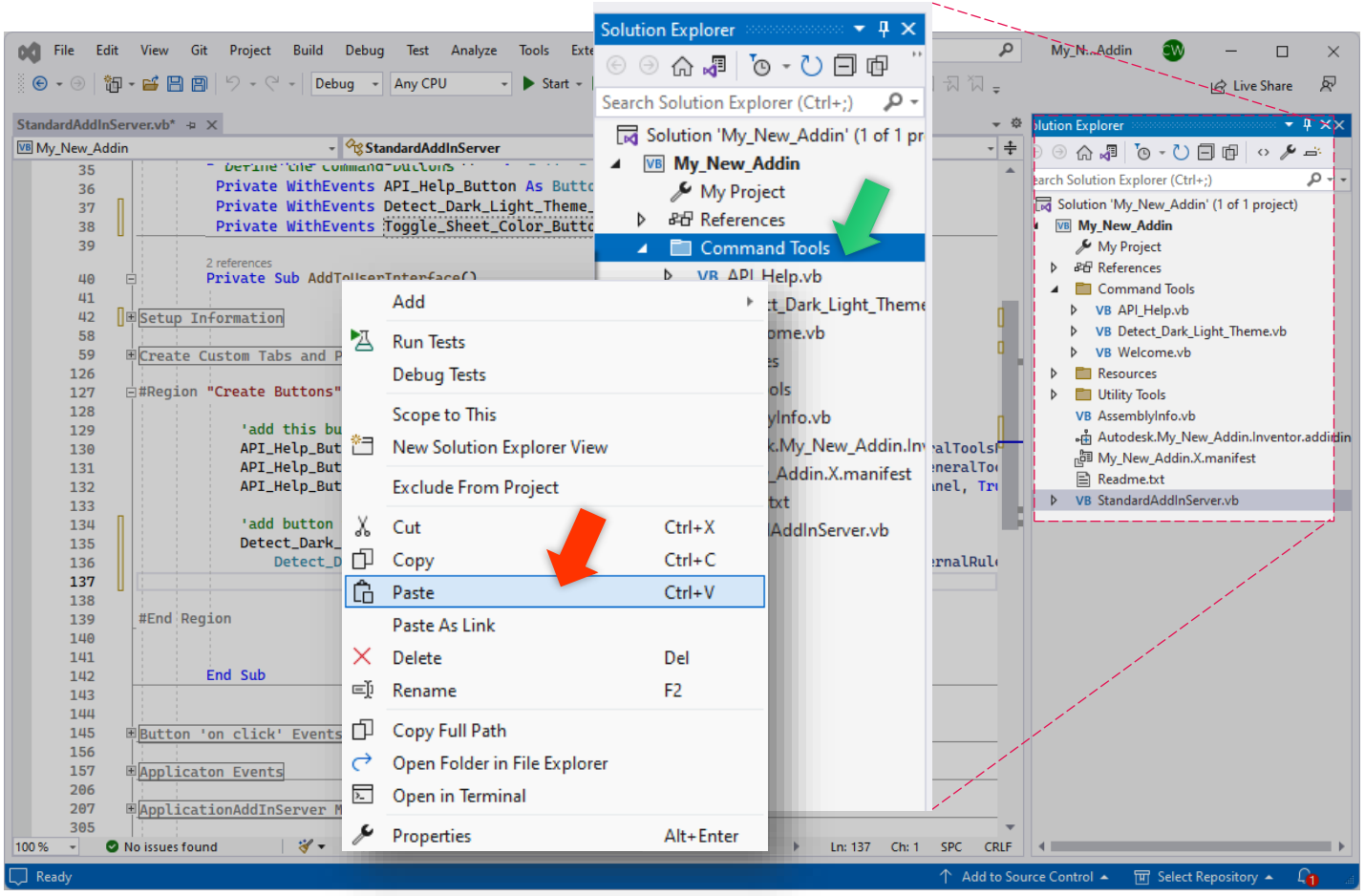
- If we expand the **Command Tools** folder, we can see the module that is being called **API_Help**

Modifying the Add-in



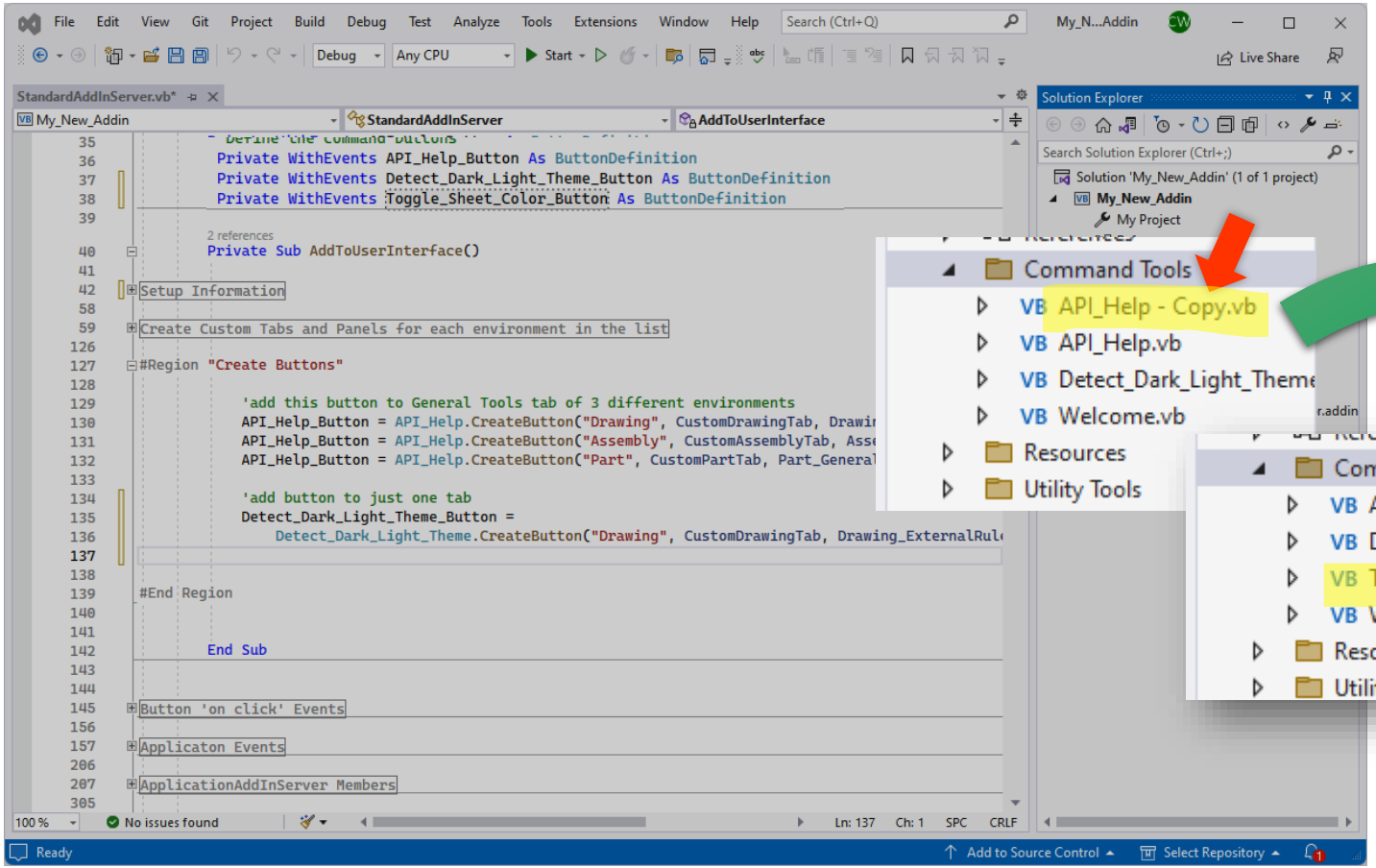
- To create a new button module we can simply copy an existing module
- Right click on the module to copy and choose Copy as shown

Modifying the Add-in



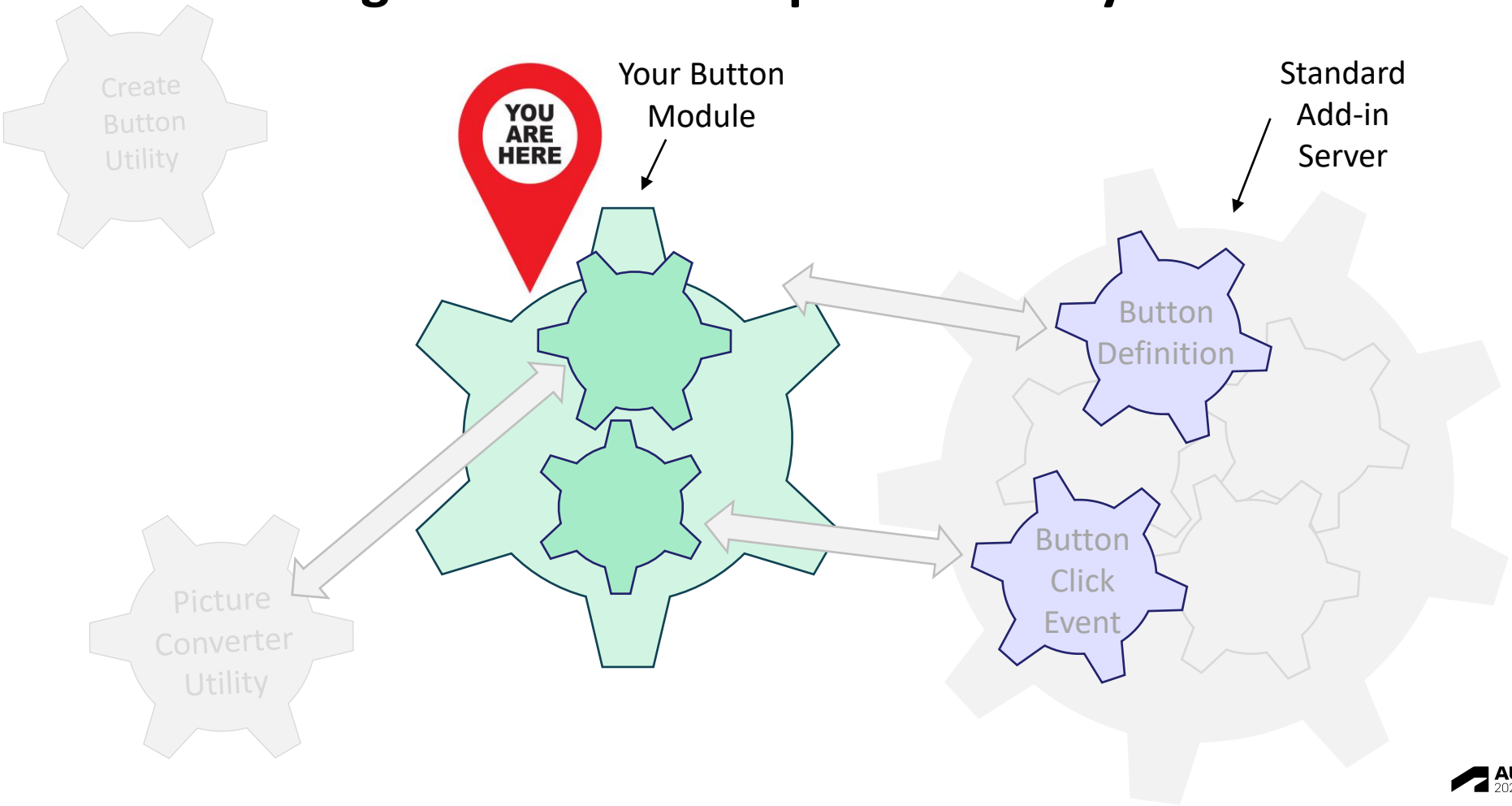
- Right click on the Command Tools folder and choose Paste

Modifying the Add-in

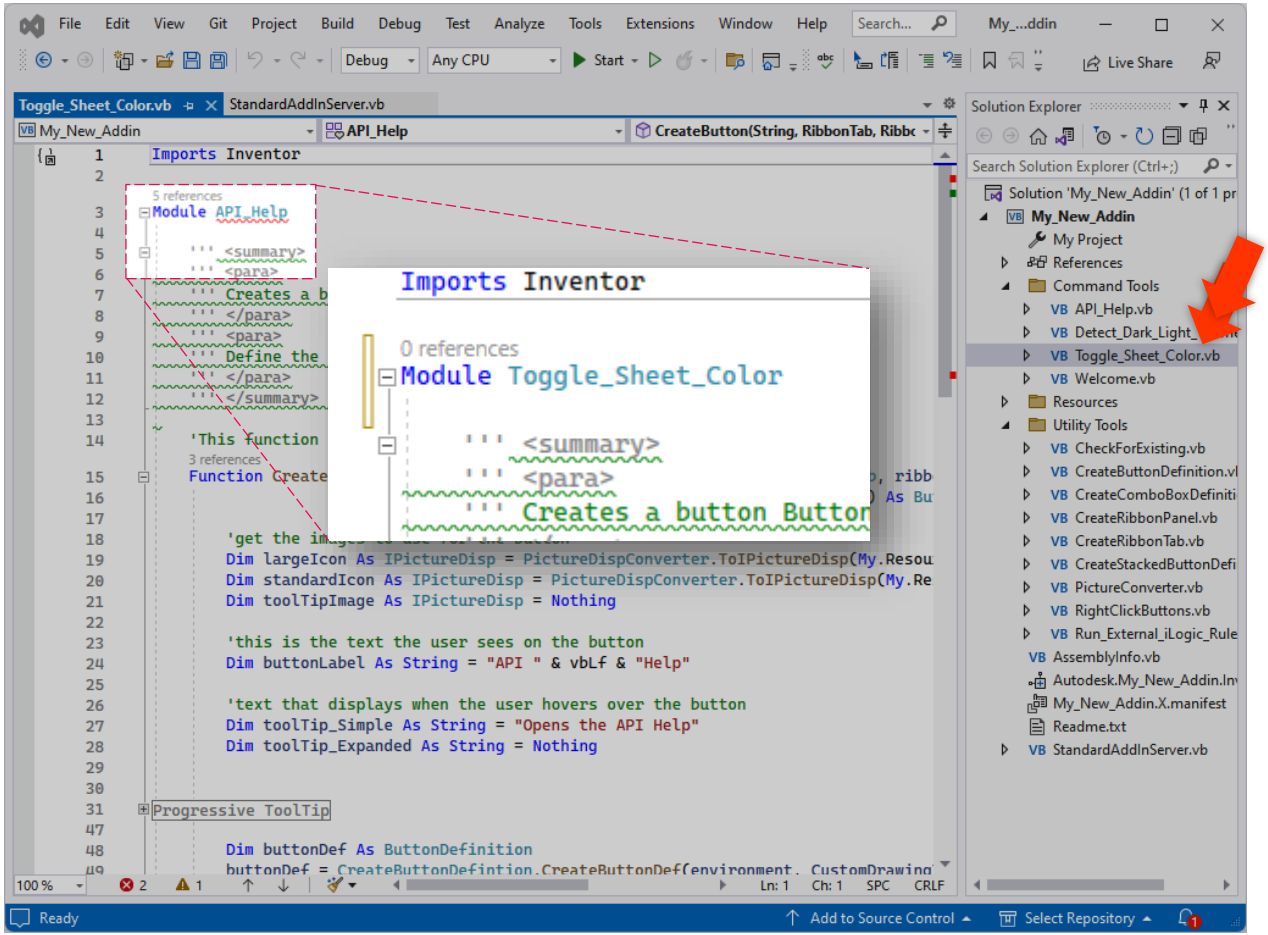


- Rename the copy

Understanding the Add-in Template Visually

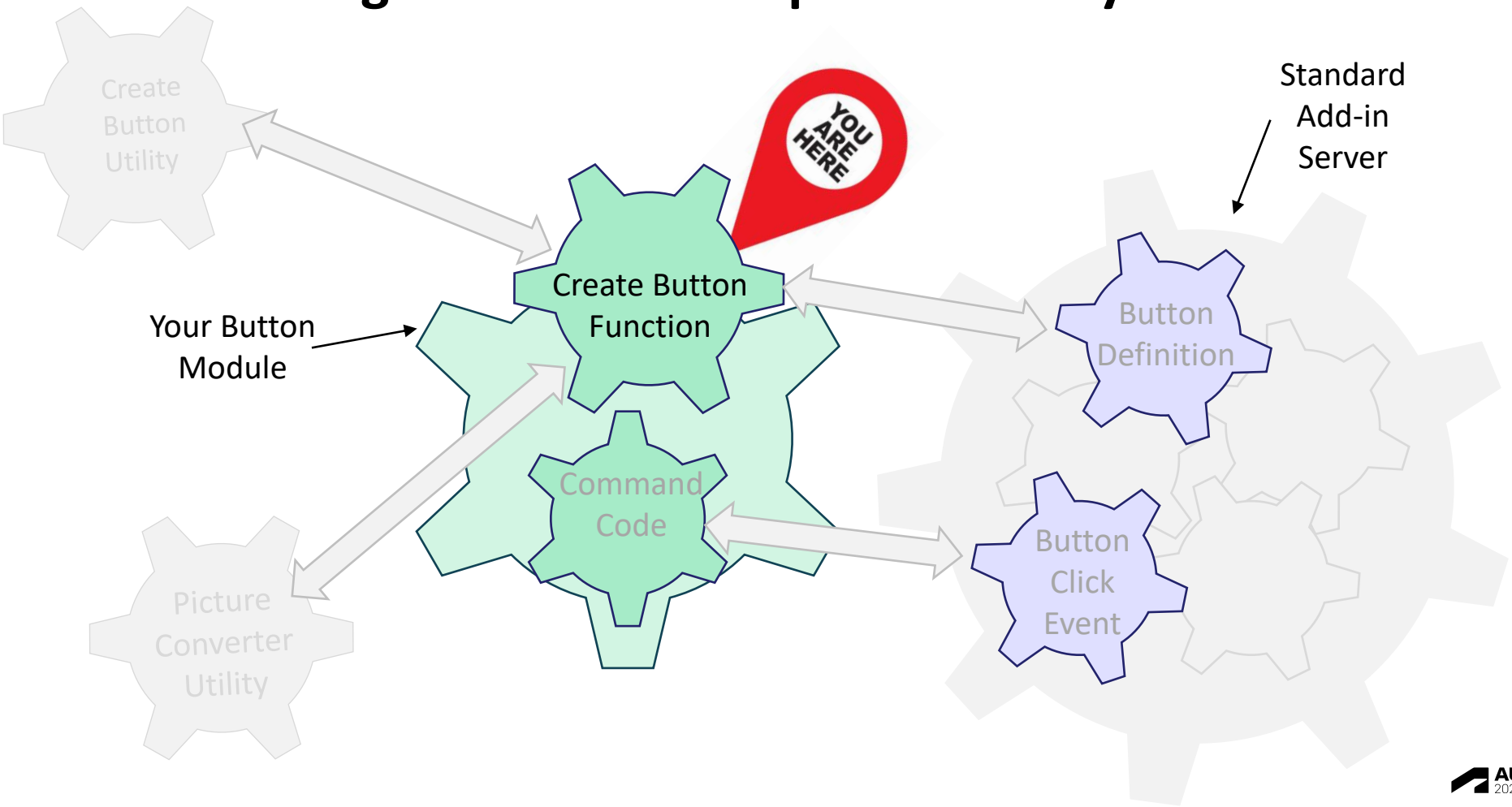


Modifying the Add-in

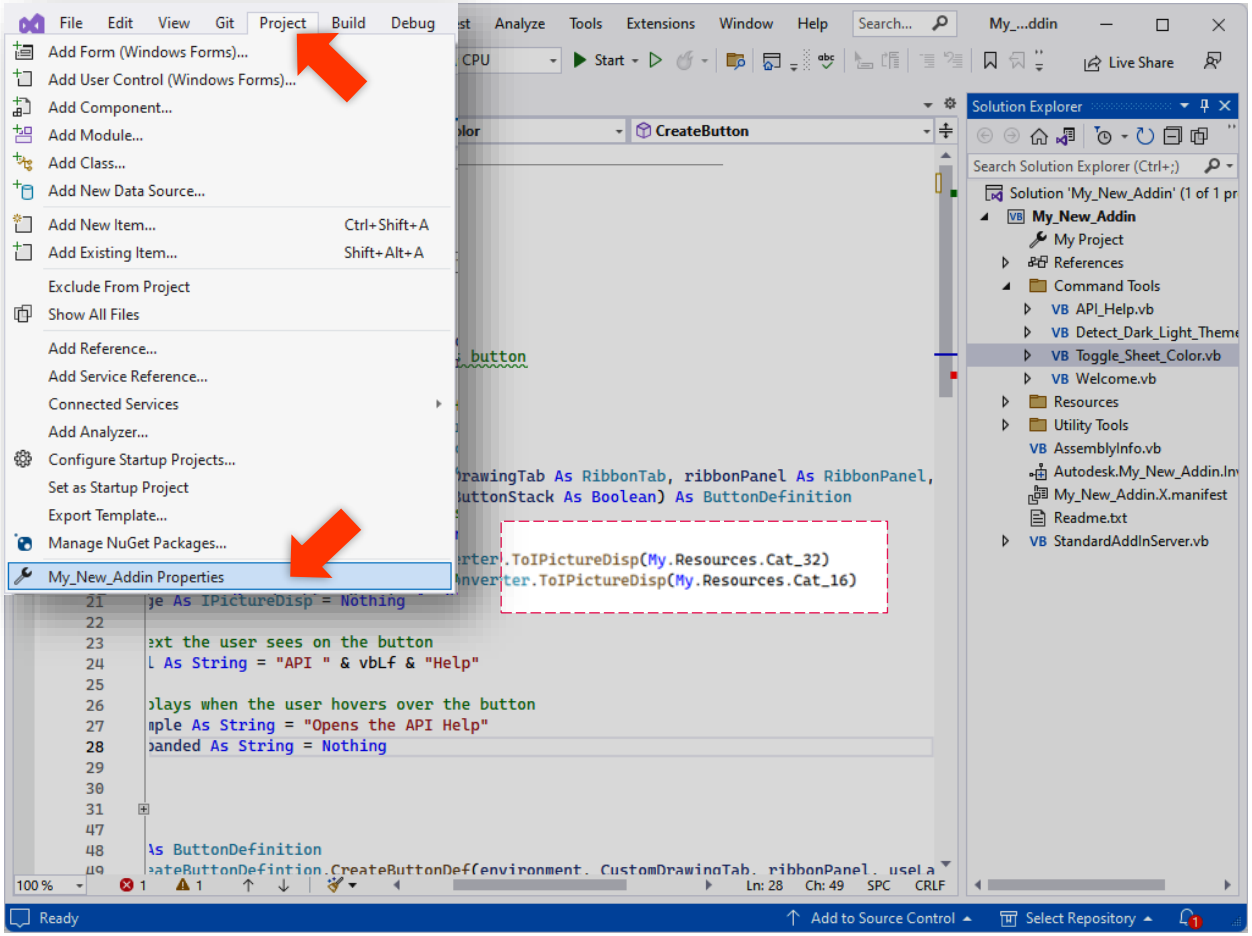


- Next we'll double click the Toggle_Sheet_Color module to open it
- And change the module name from **API_Help** to **Toggle_Sheet_Color**

Understanding the Add-in Template Visually



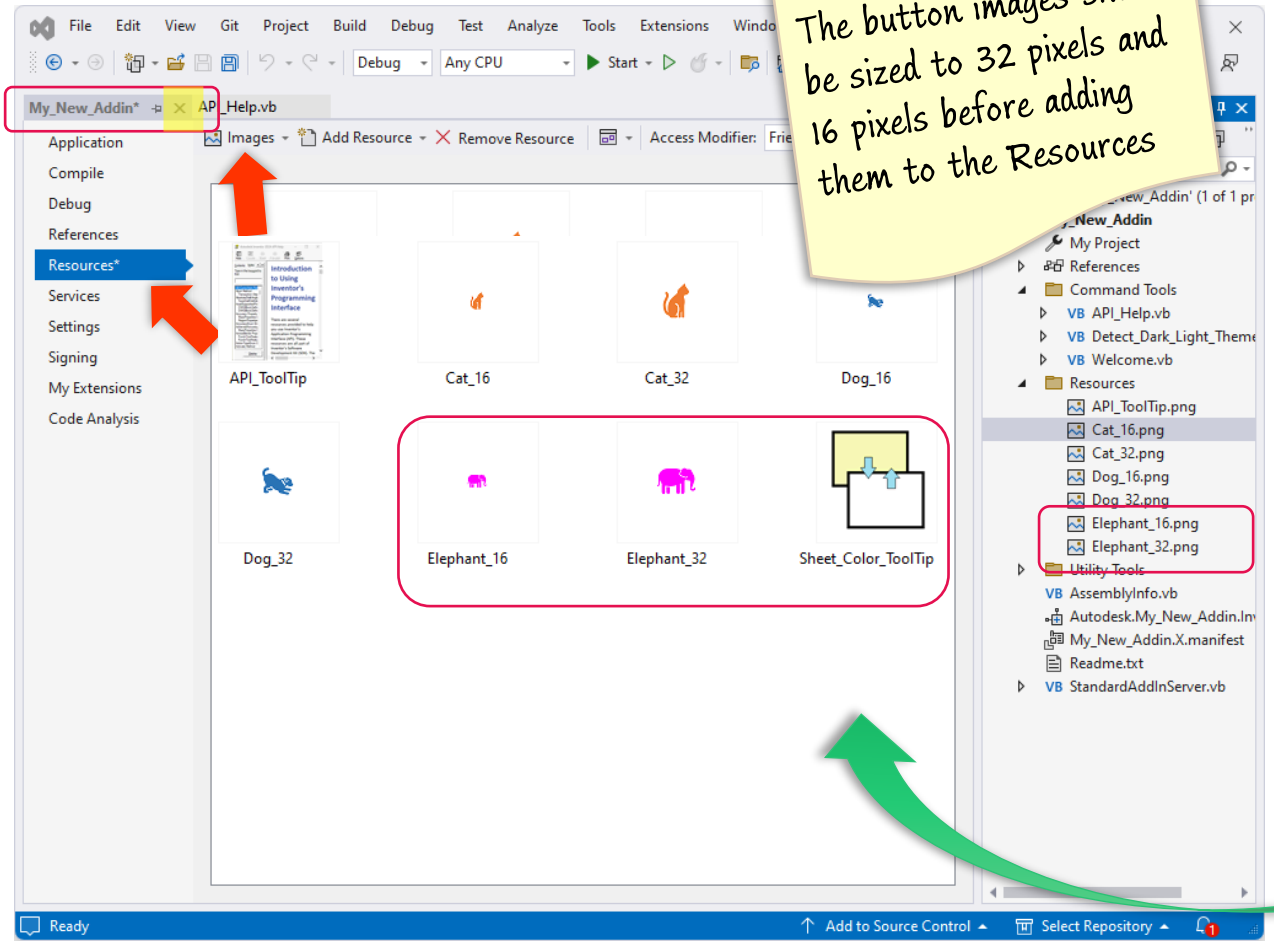
Modifying the Add-in



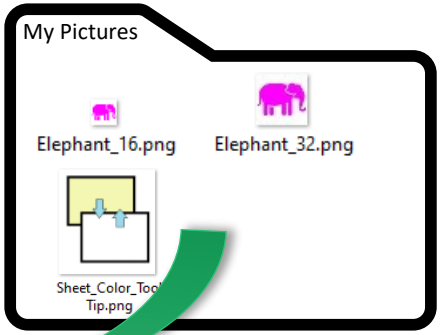
- Next we'll change the button image
- But first we need to add an image to our Resources.
- Go to **Project** > **Properties** as shown

Modifying the Add-in

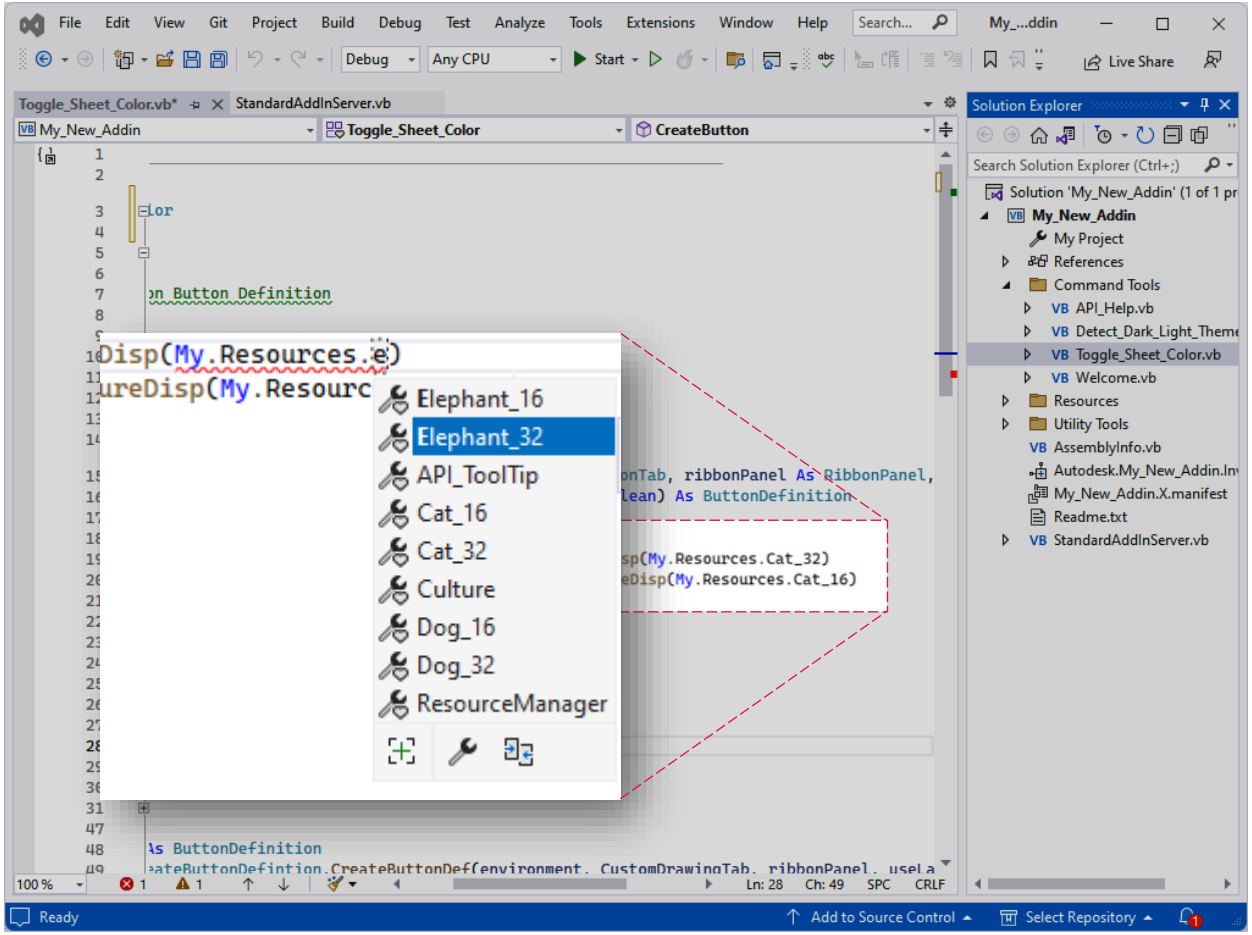
Note:
The button images should be sized to 32 pixels and 16 pixels before adding them to the Resources



- You can use your own images or use the ones provided with the download materials
- Simply drag and drop the image on the Images pane as shown
- Then click the X to close and save the **My_New_Addin** file

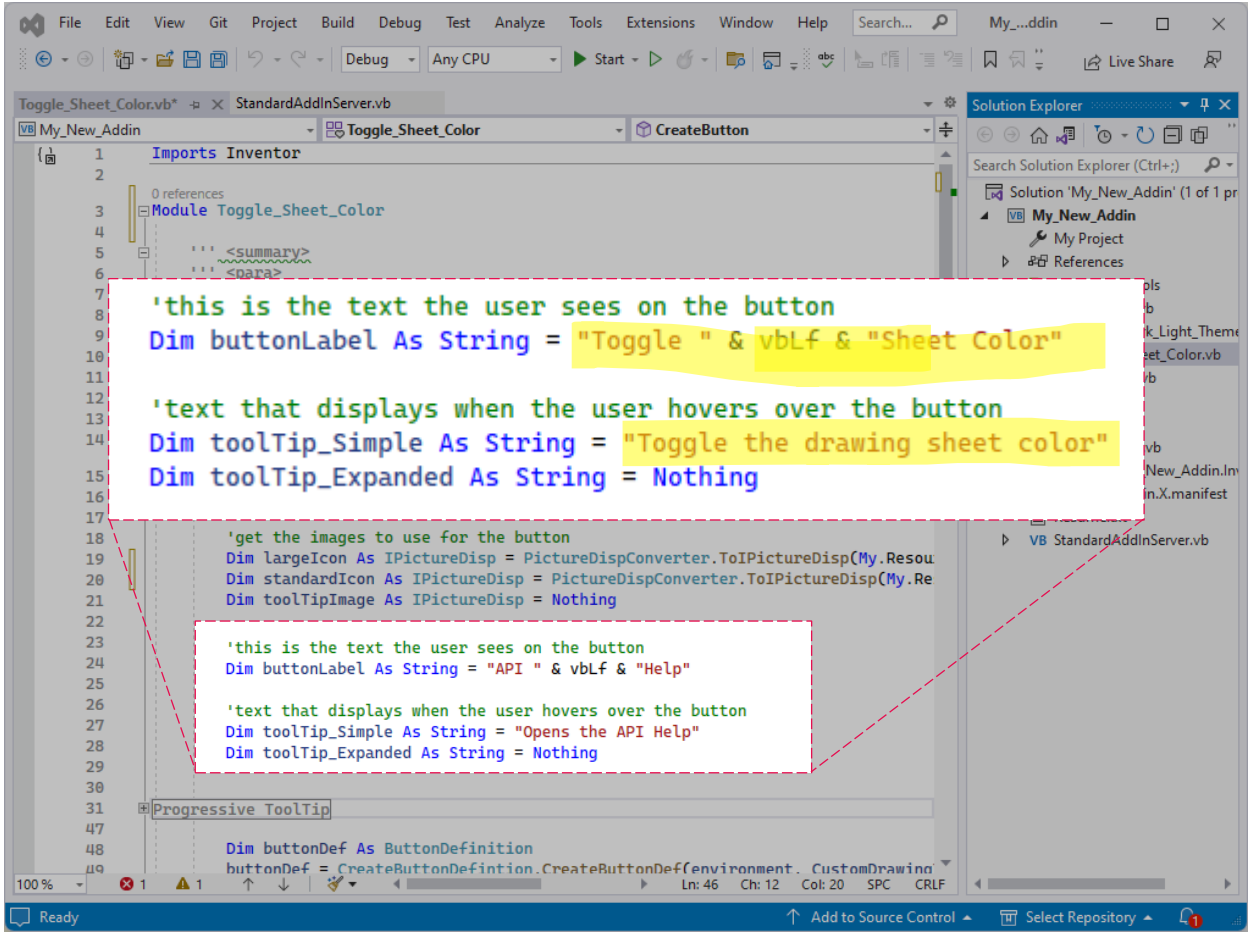


Modifying the Add-in



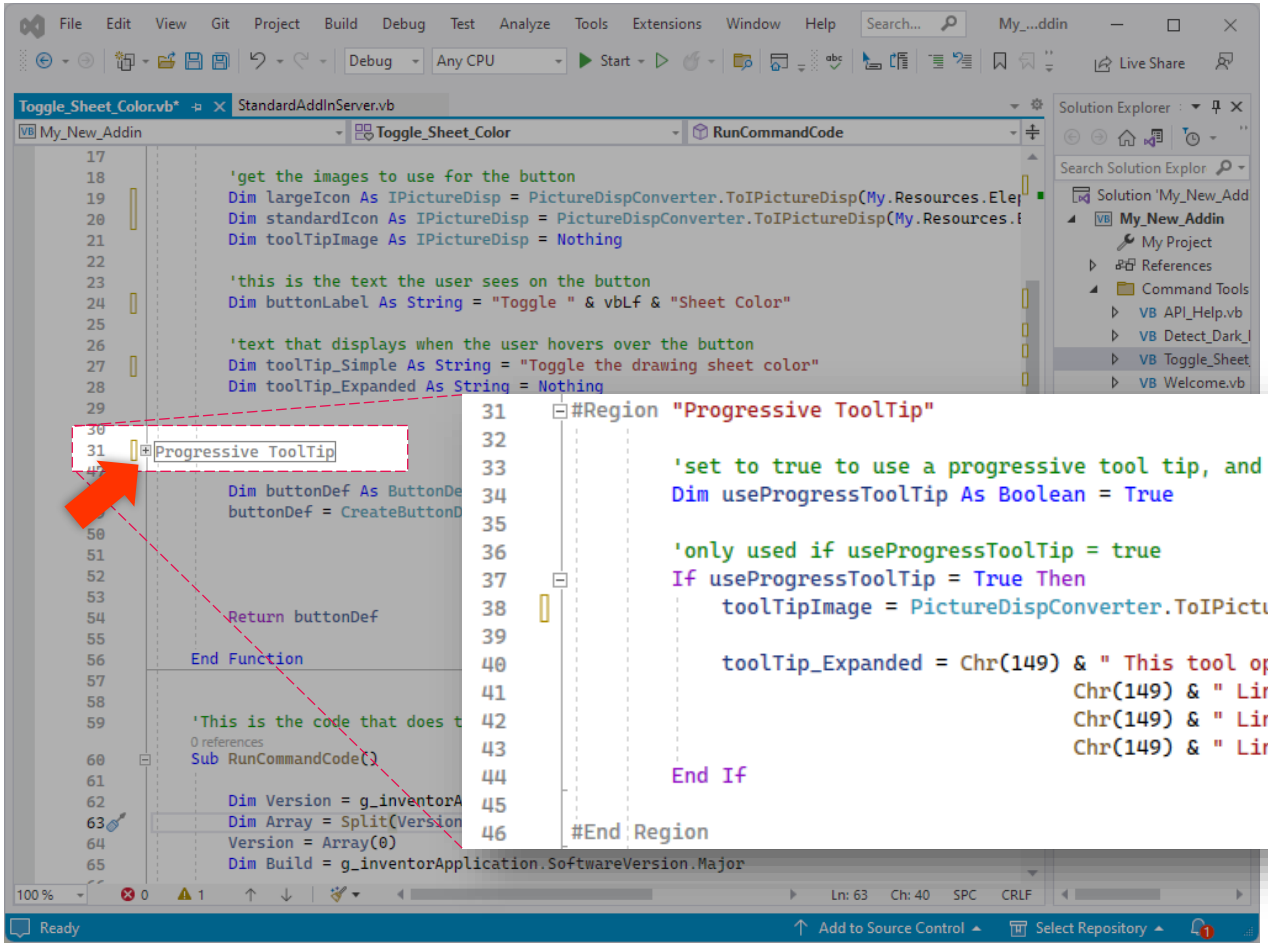
- Returning to our new command module file…
- Backspace over Cat_32 and start typing the name of your images to see them appear in the list
- You can then just select the image from the list and double-click it
- Do the same to the _16 image

Modifying the Add-in



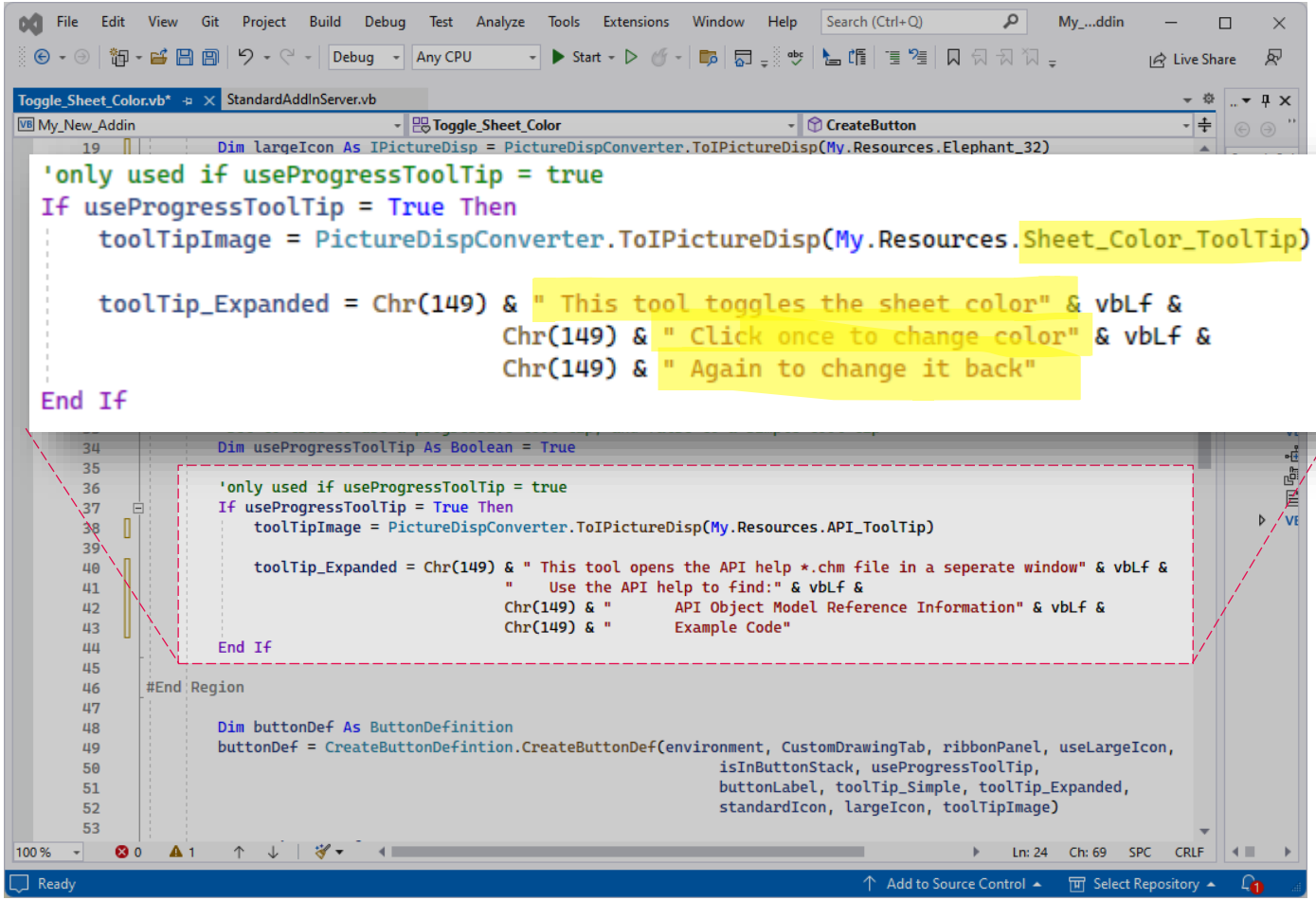
- Next we'll update the button label
- And the simple tool tip string

Modifying the Add-in



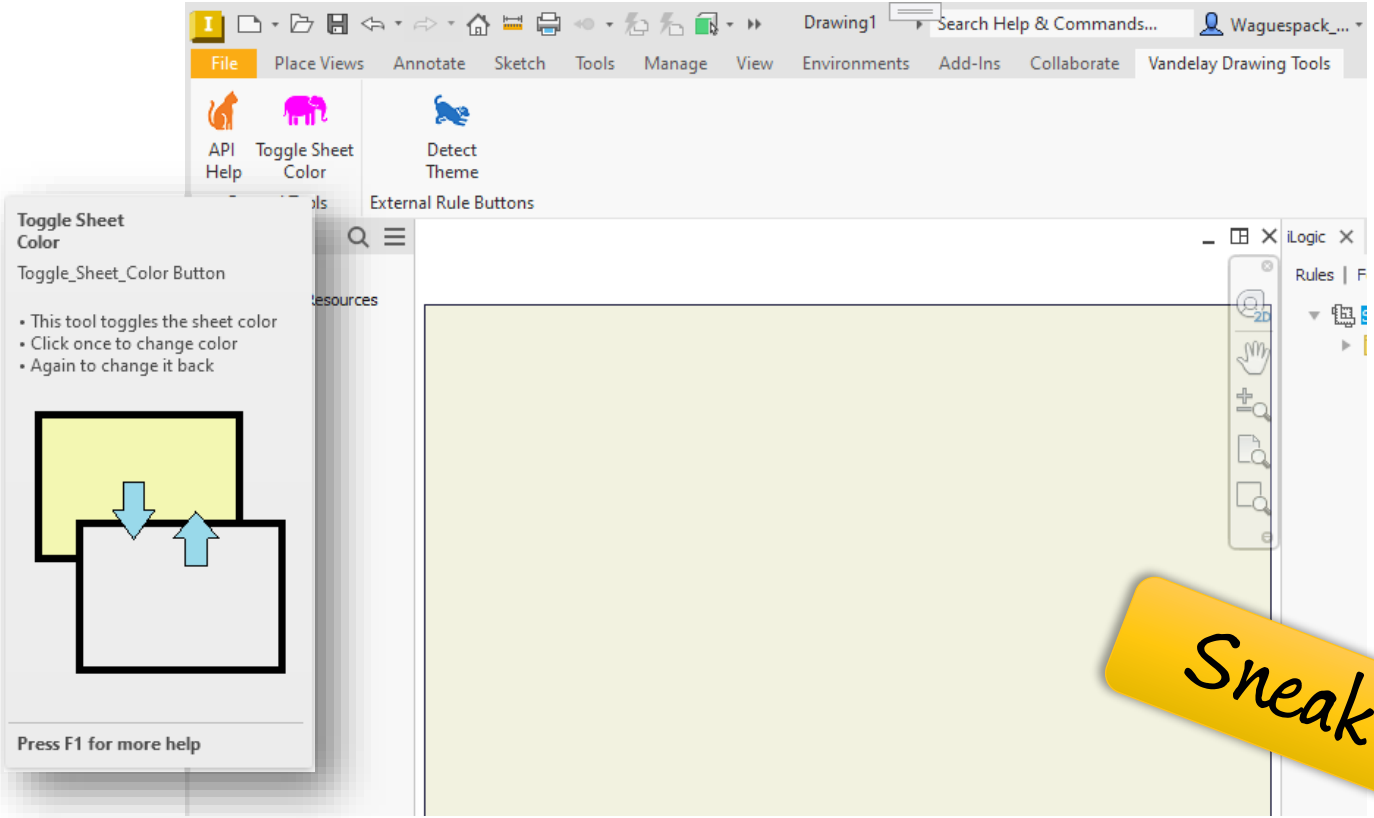
- Next we'll expand the Progressive ToolTip region by clicking the + sign

Modifying the Add-in



- We'll change the tooltip image
- As well as the extended tooltip text

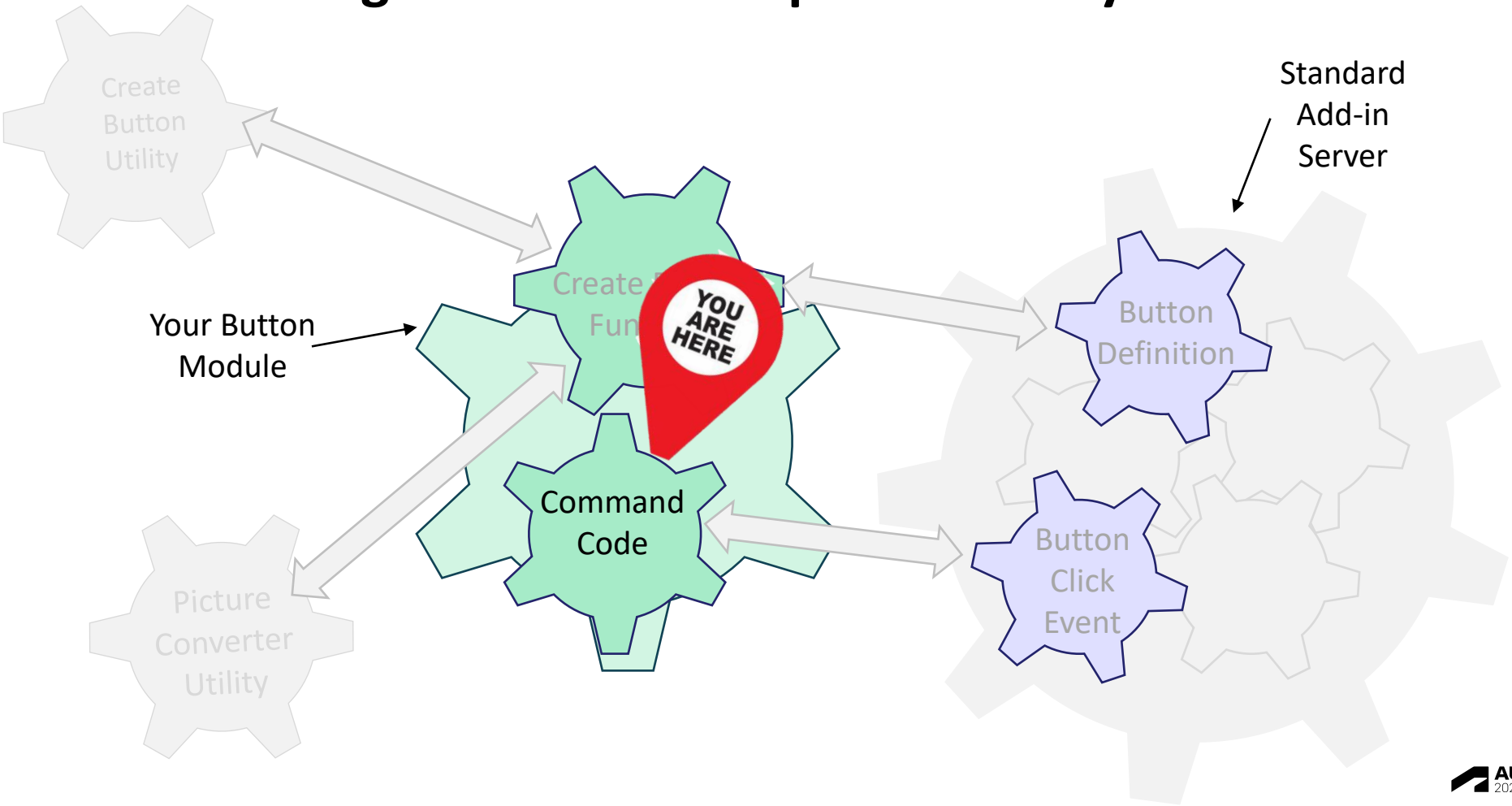
Modifying the Add-in



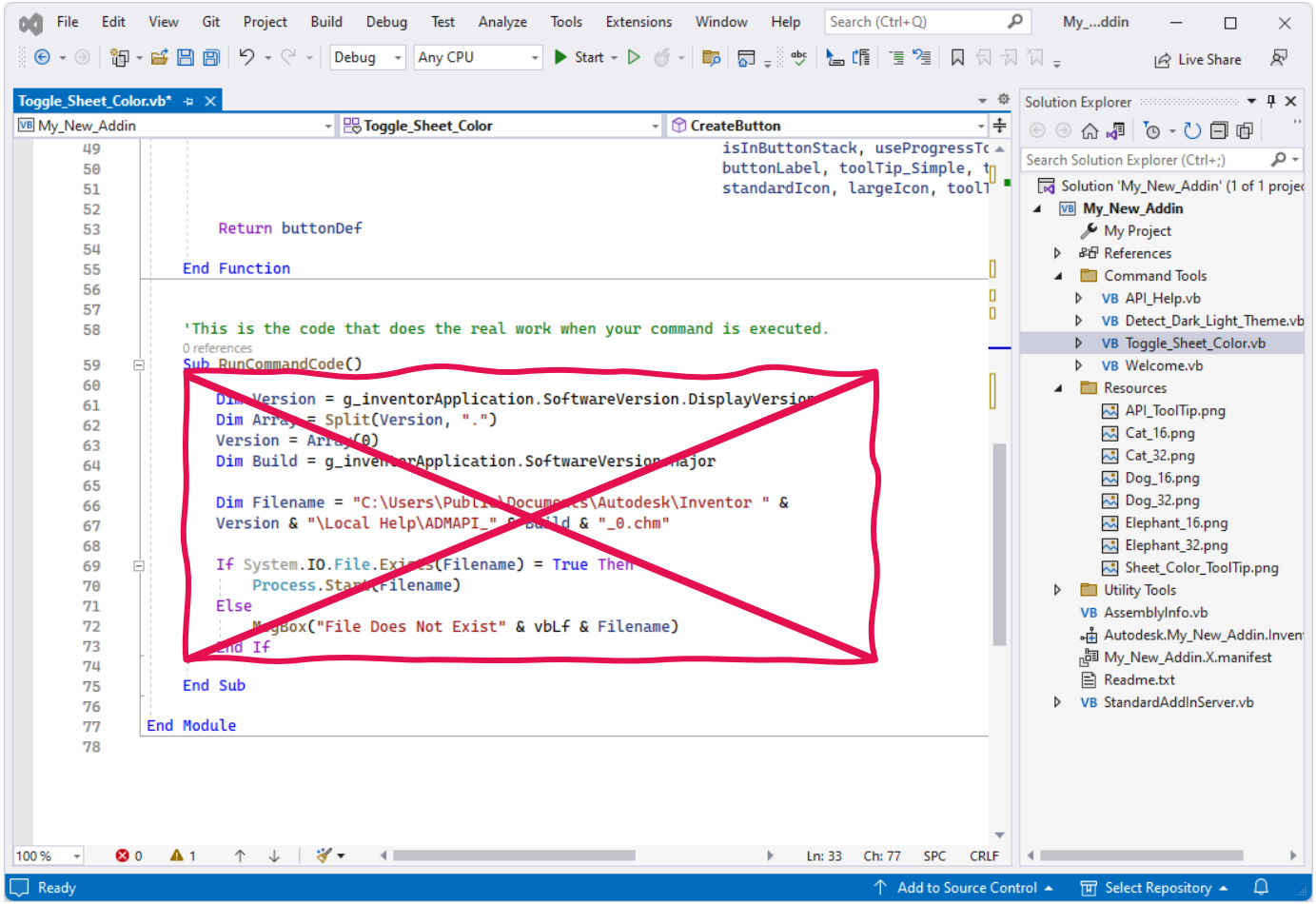
Sneak Peak

- This is what the extended tooltip that we are creating will look like

Understanding the Add-in Template Visually

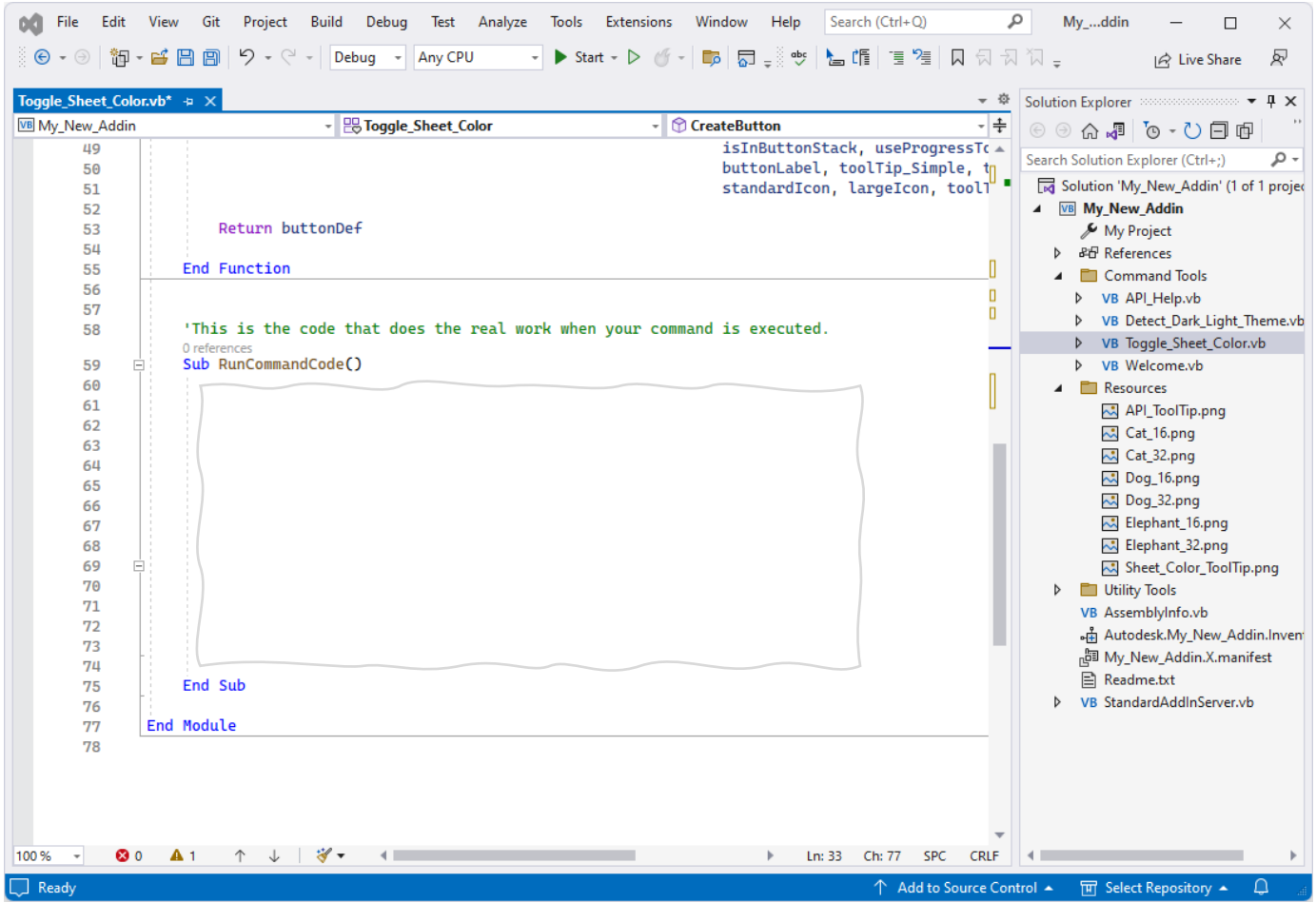


Modifying the Add-in



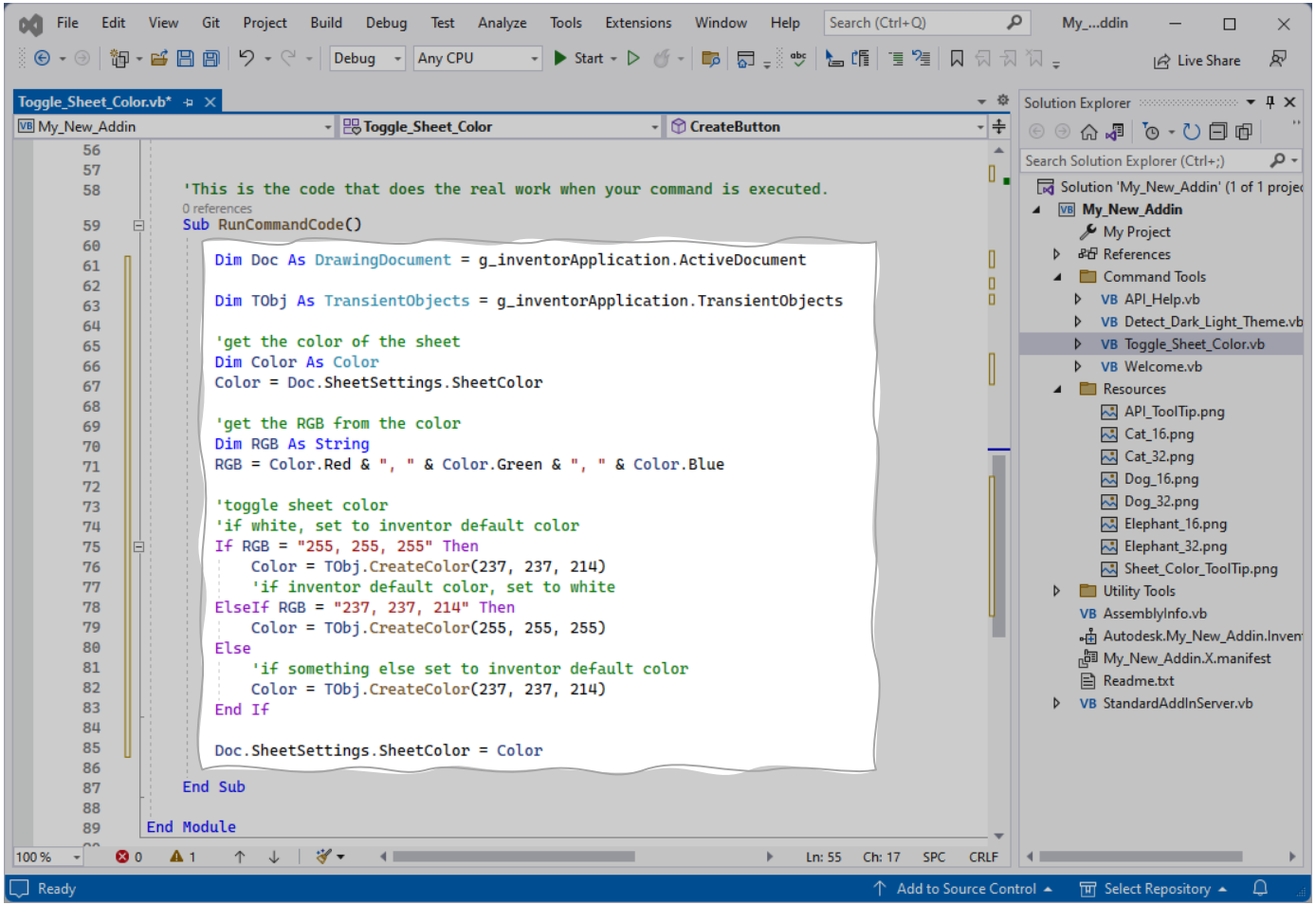
- Next we'll scroll down to the **RunCommandCode** sub
- Delete all of the code between the **Sub** and **End Sub** lines

Modifying the Add-in



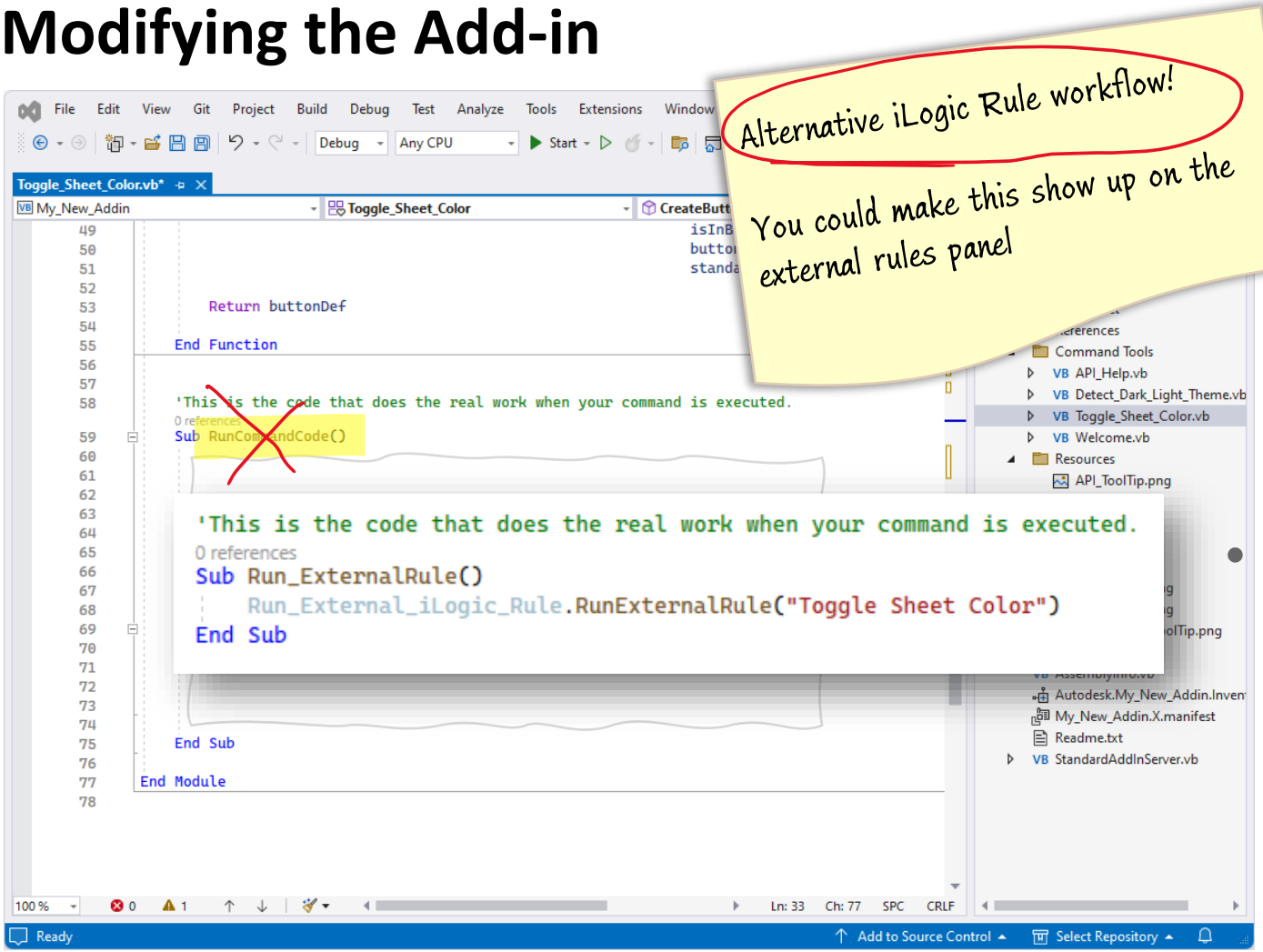
- Next we'll scroll down to the **RunCommandCode** sub
- Delete all of the code between the **Sub** and **End Sub** lines

Modifying the Add-in



- Add the code that we want to run when the button is clicked.
- You can find this example code in the download materials for this class

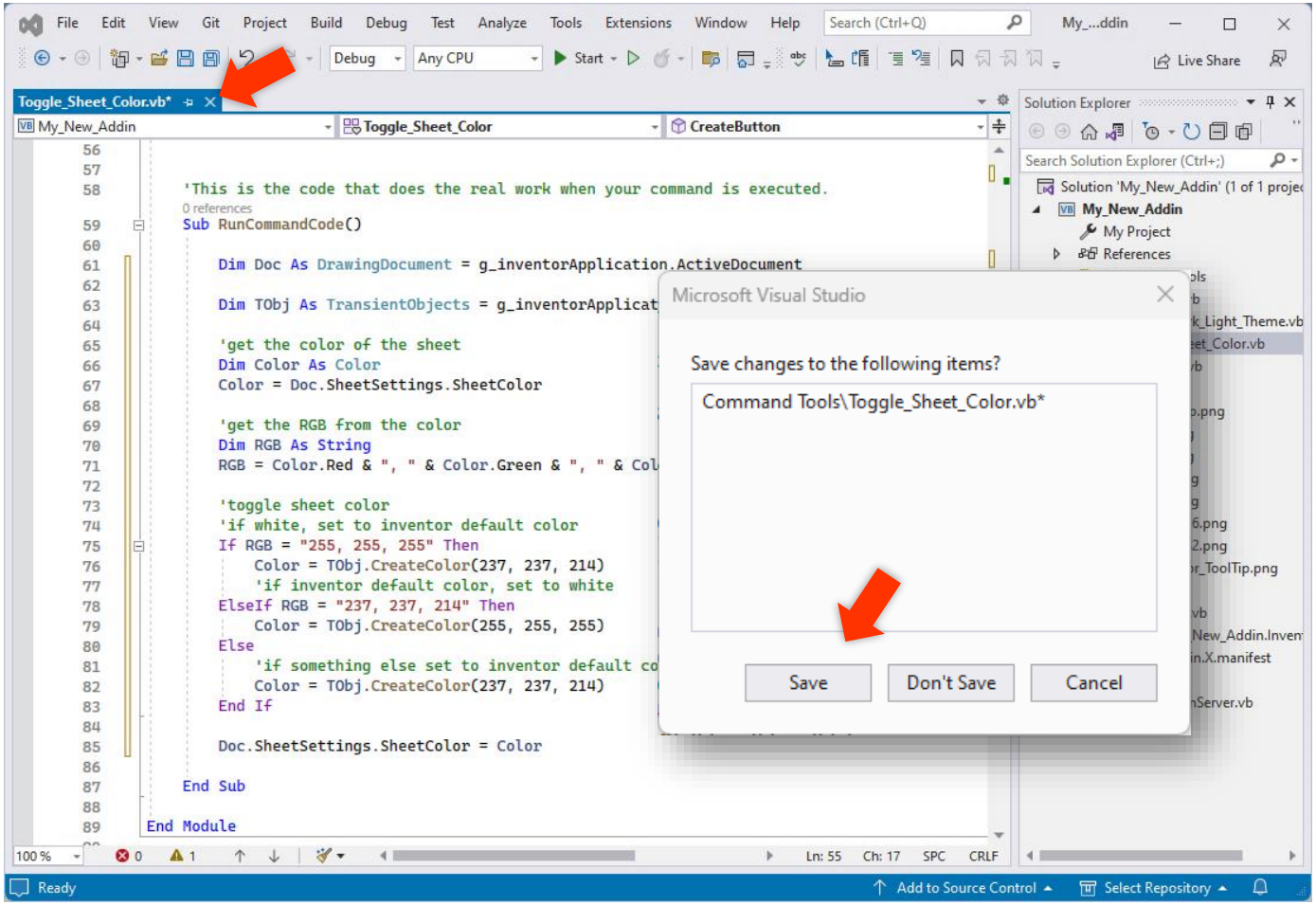
Modifying the Add-in



you wanted this module to run an external iLogic rule, rather than executing code from the add-in, then you would change the sub as shown.

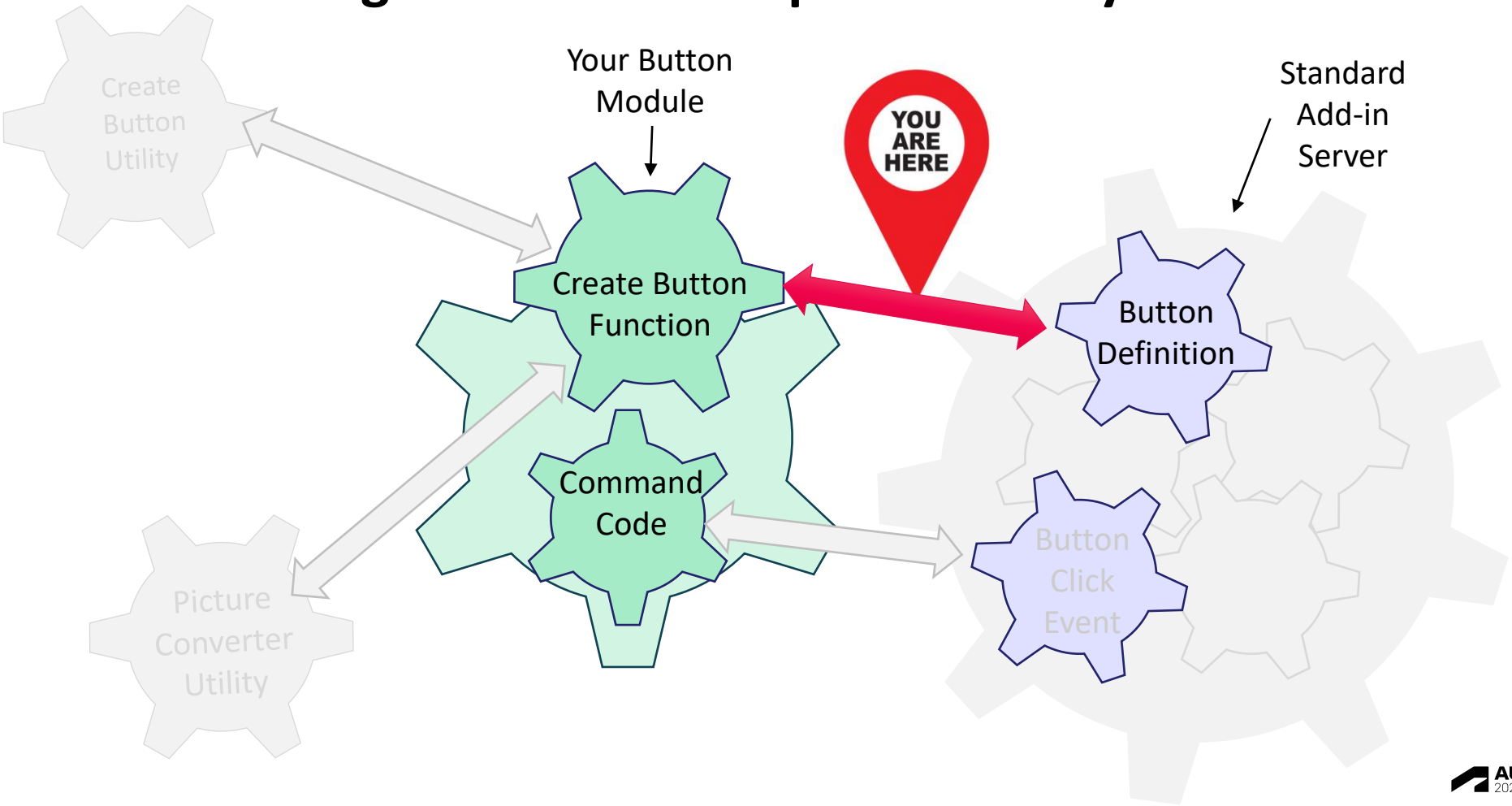
This will use a utility module called **Run_External_iLogic_Rule** as shown

Modifying the Add-in

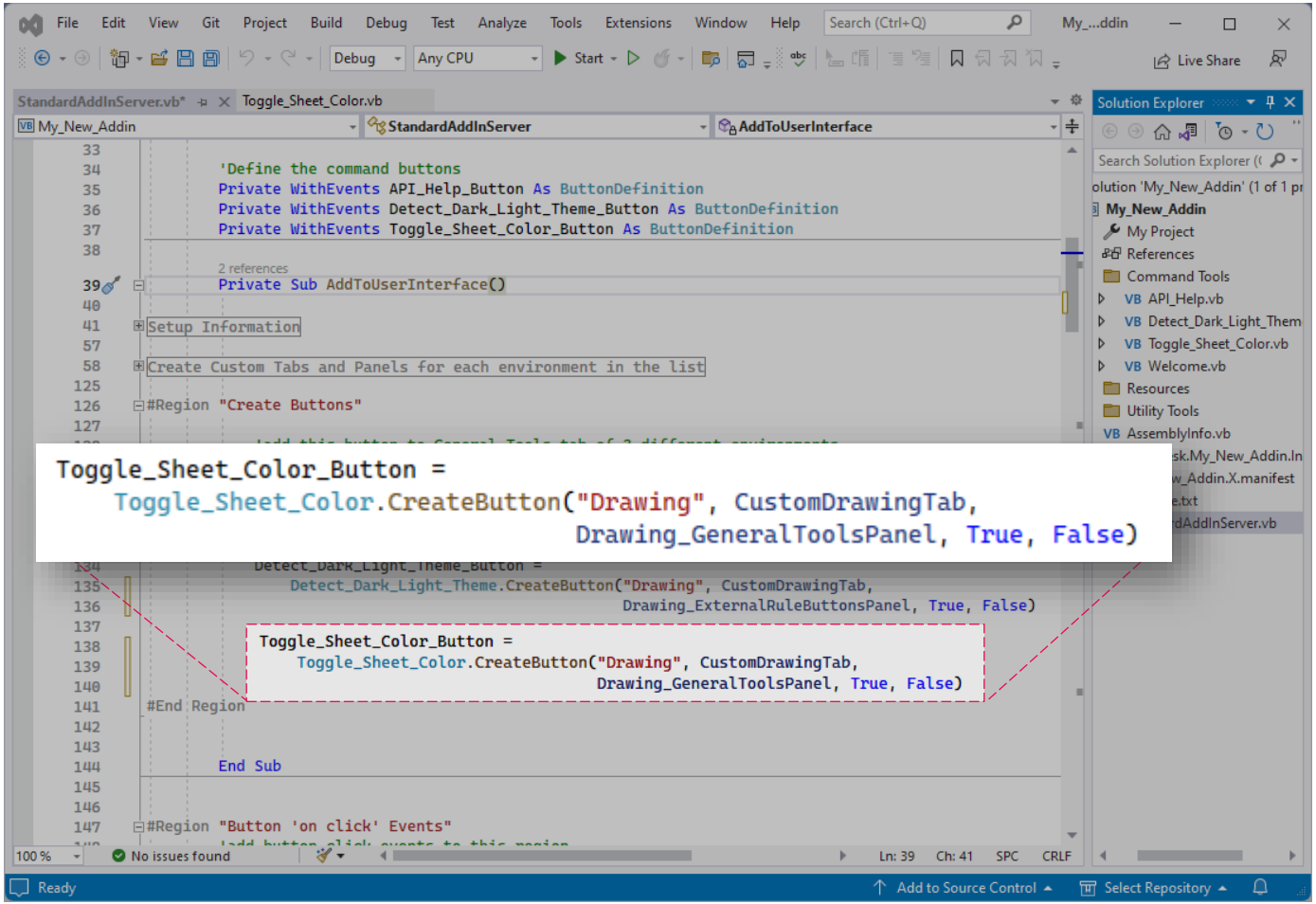


- We're done!
- We've just created a new button module.
- Next, we'll "wire it up" in the StandardAddInServer file.
- But fist click the close **X** and then click the **Save** button, as shown

Understanding the Add-in Template Visually

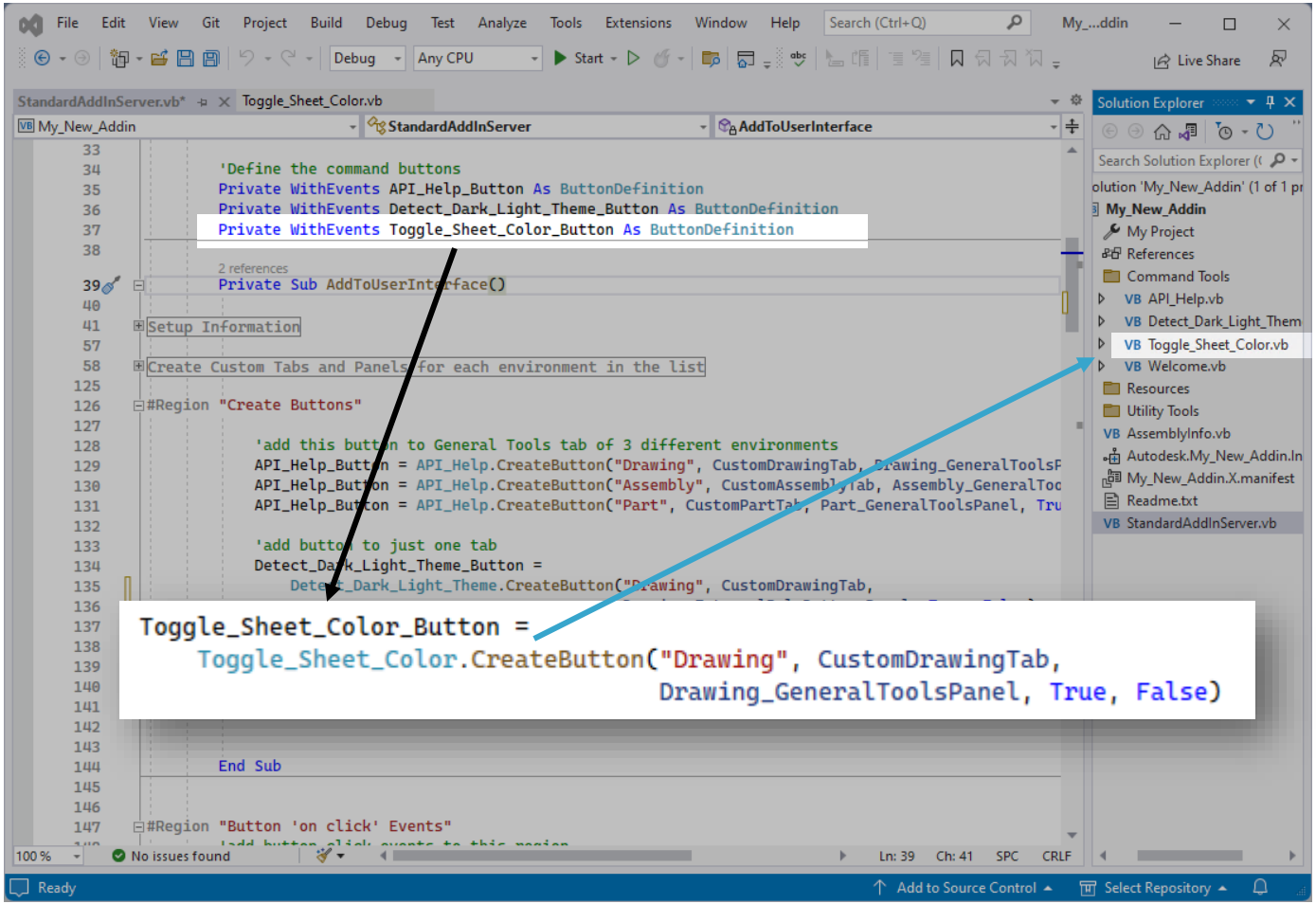


Modifying the Add-in



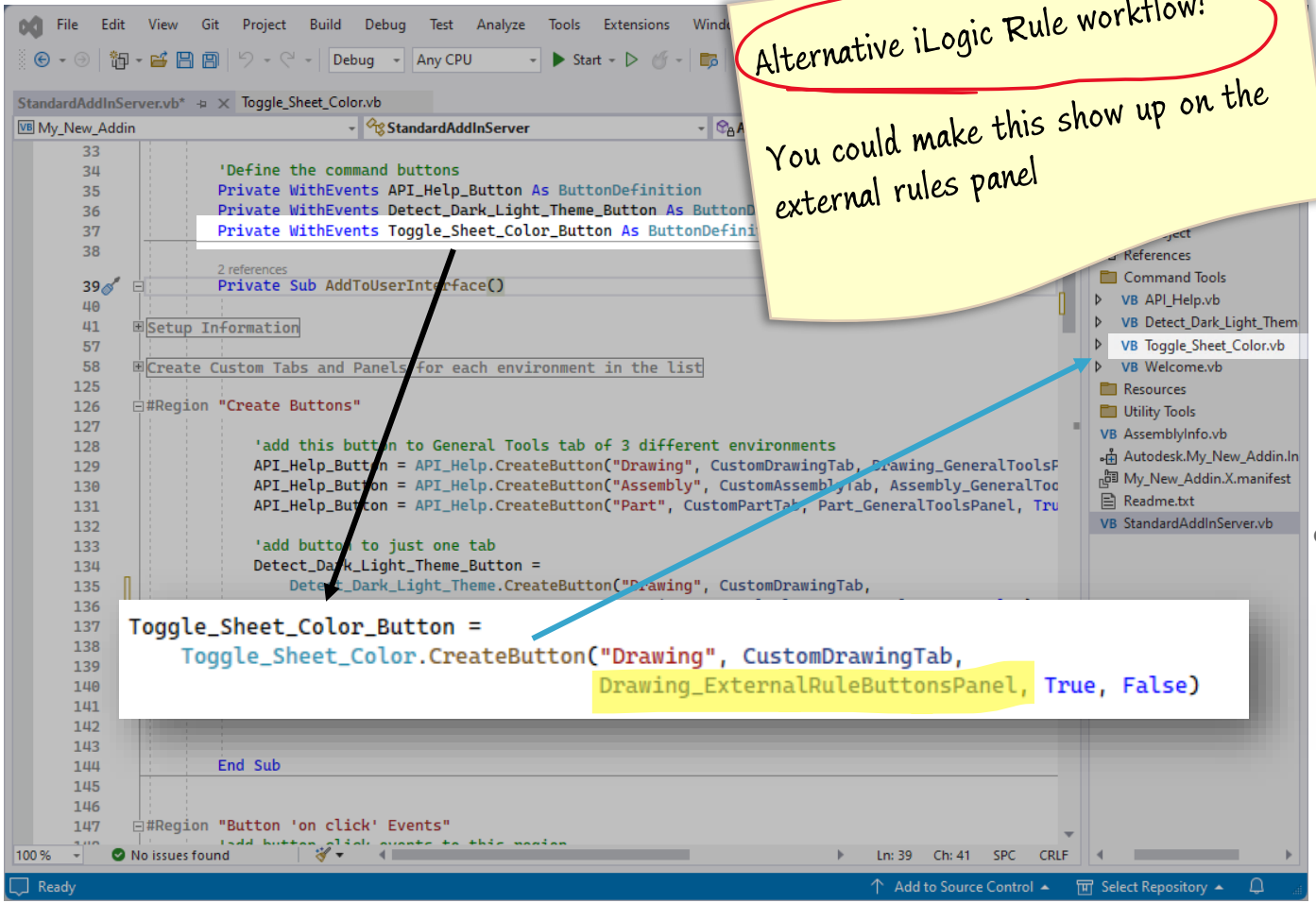
- In the Create Buttons region of the StandardAddinServer…
- We'll add a line to connect the button definition we created earlier, with the Function in the module we just created.
- This line of code is run when Inventor starts up and loads the add-in

Modifying the Add-in



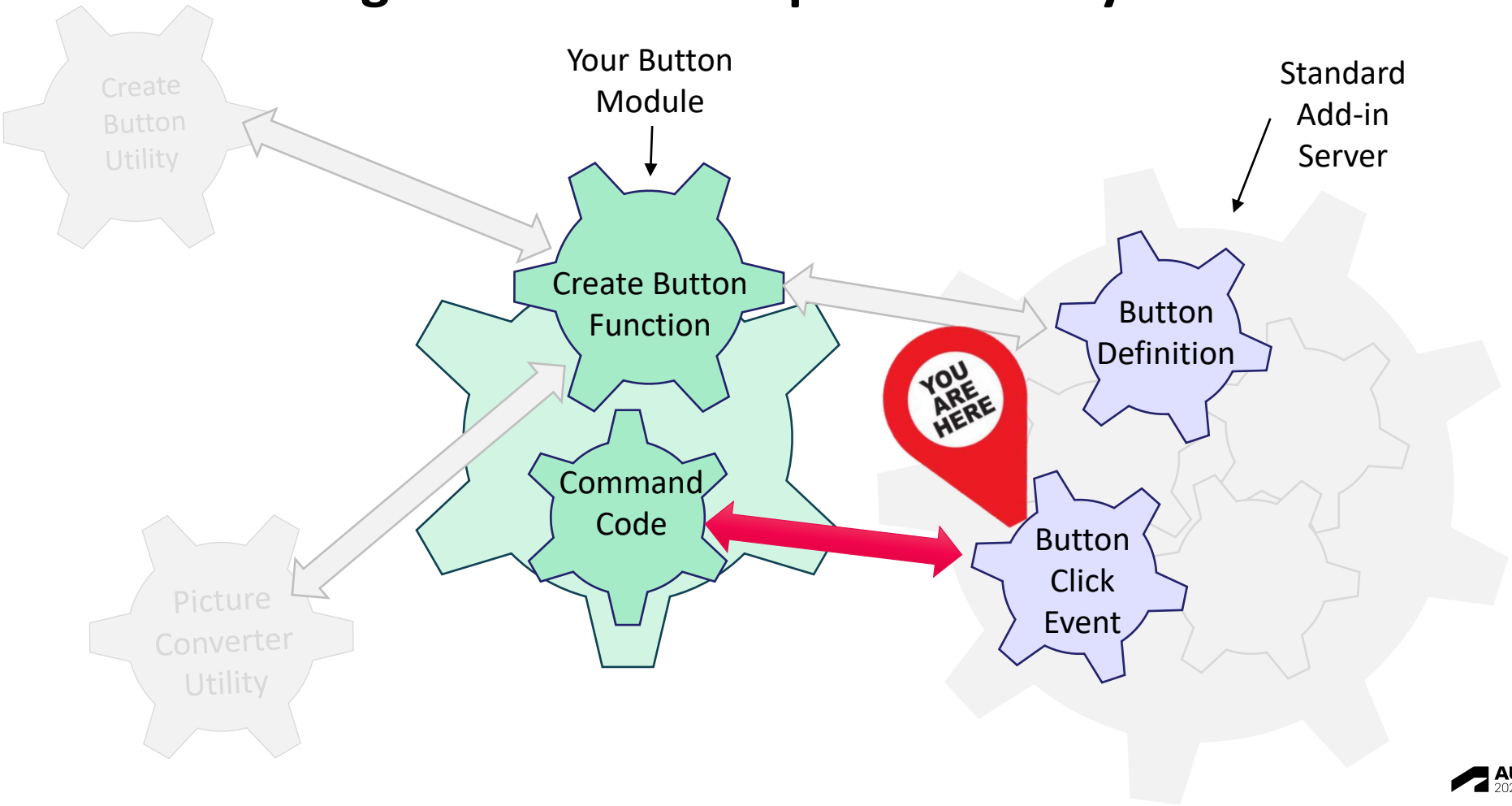
- We'll add a line to connect the button definition we created earlier, with the Function in the module we just created.
- This line of code is run when Inventor starts up and loads the add-in

Modifying the Add-in

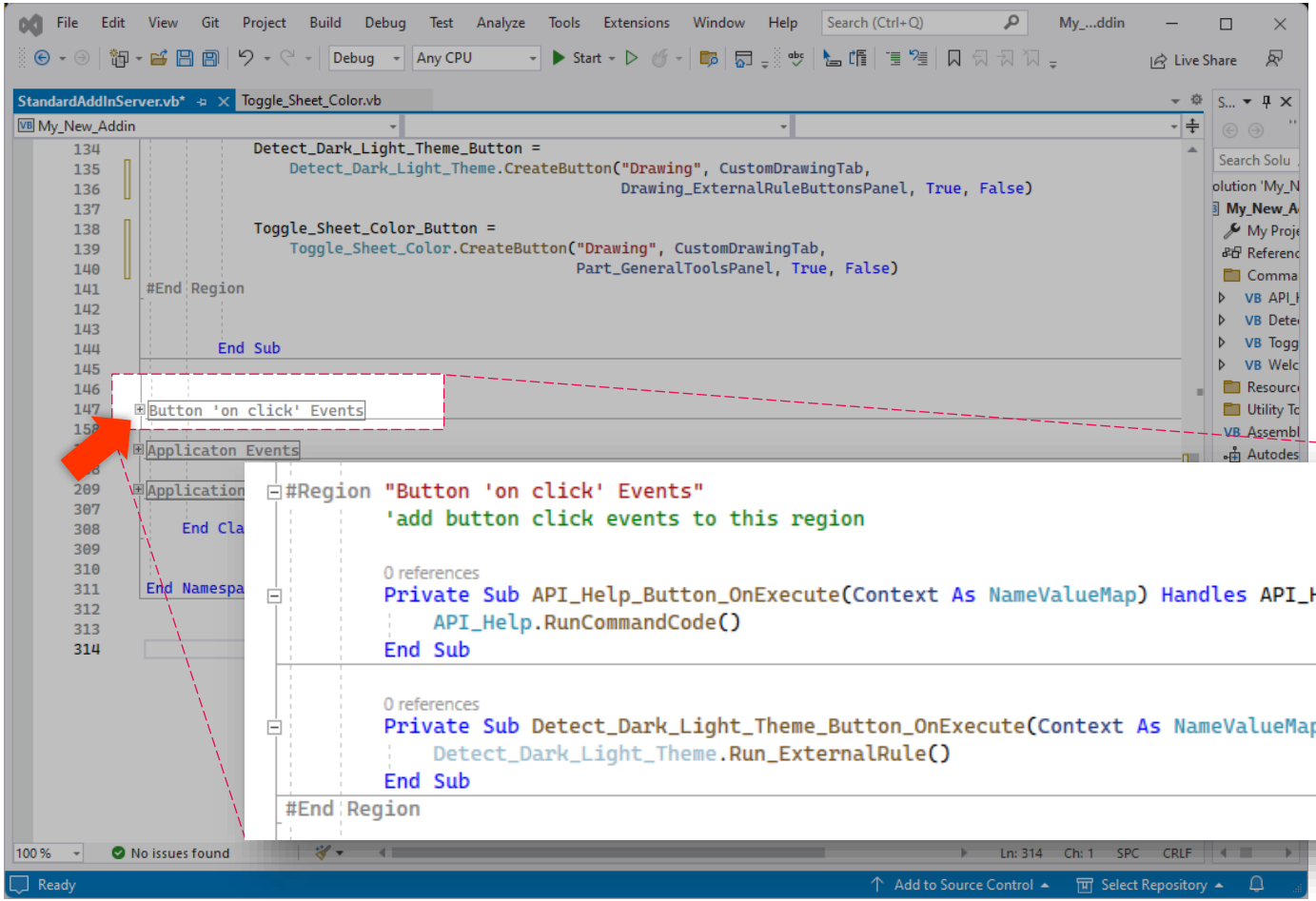


- We'll add a line to connect the button definition we created earlier, with the Function in the module we just created.
- This line of code is run when Inventor starts up and loads the add-in

Understanding the Add-in Template Visually

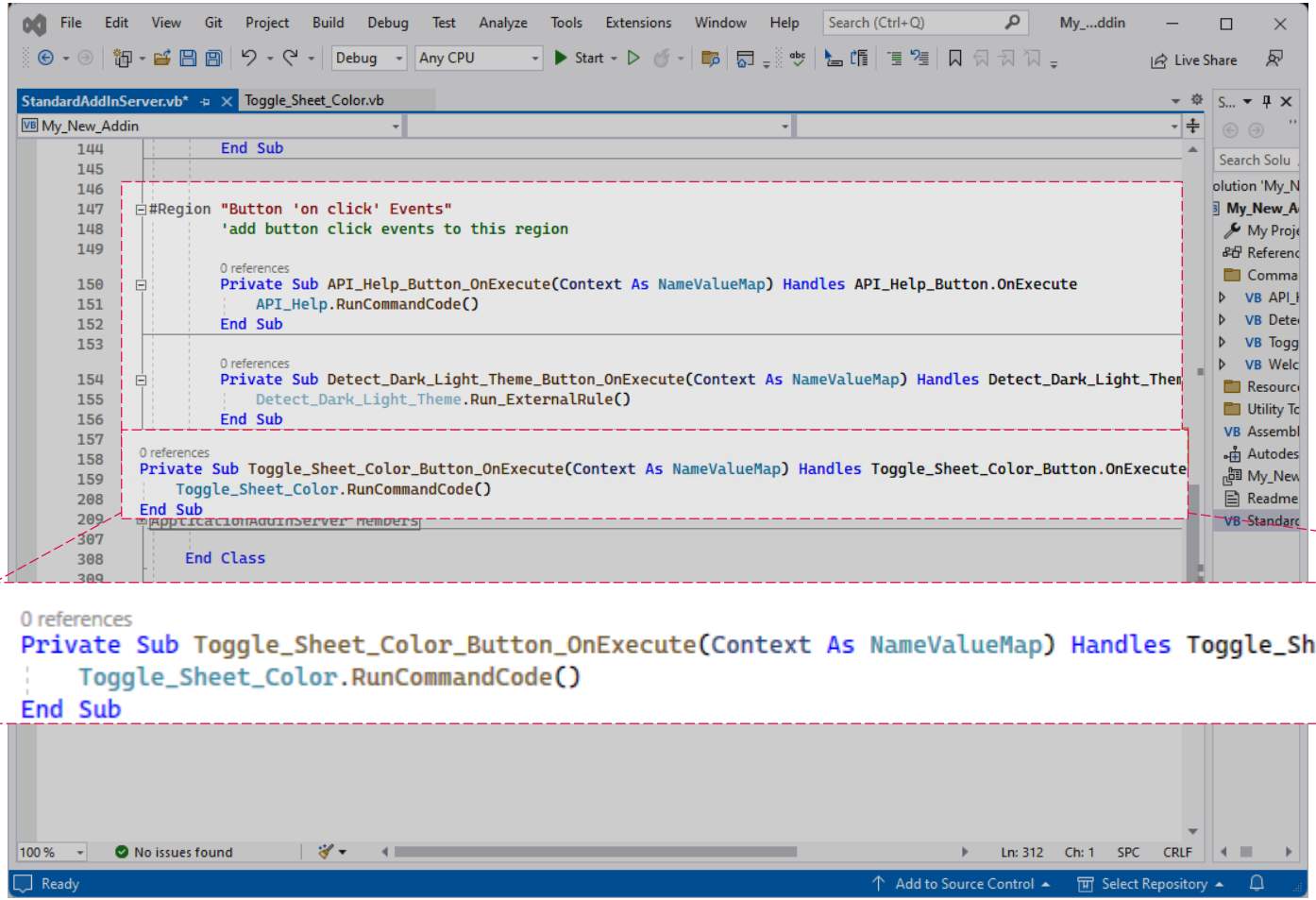


Modifying the Add-in



- Next we'll expand the **Button 'on click'** region by clicking the **+** sign

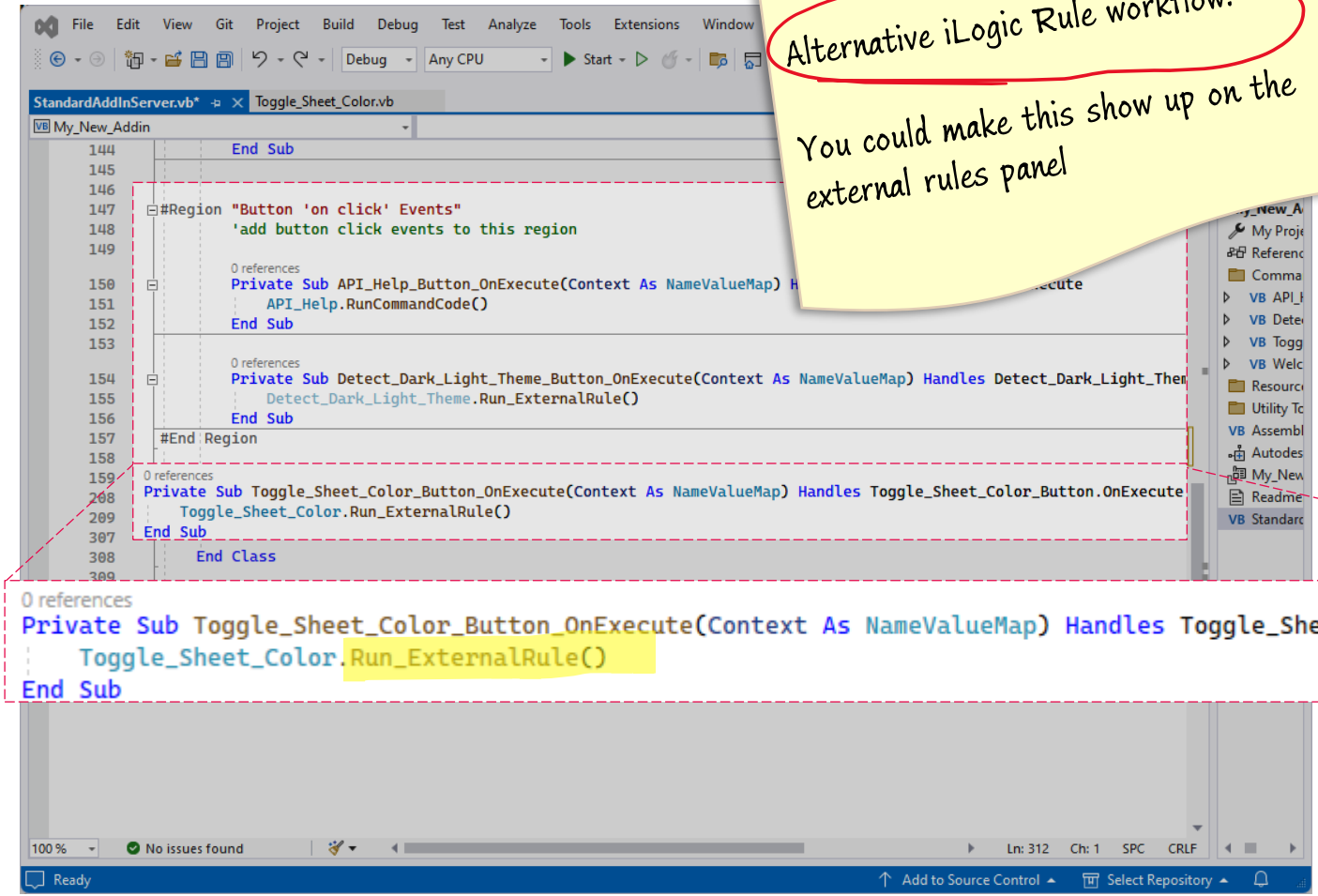
Modifying the Add-in




```
0 references
Private Sub Toggle_Sheet_Color_Button_OnExecute(Context As NameValueMap) Handles Toggle_Sheet_Color_Button.OnExecute
    Toggle_Sheet_Color.RunCommandCode()
End Sub
```

- Add a sub to run the Toggle_Sheet_Color Module command code wen the button is executed

Modifying the Add-in



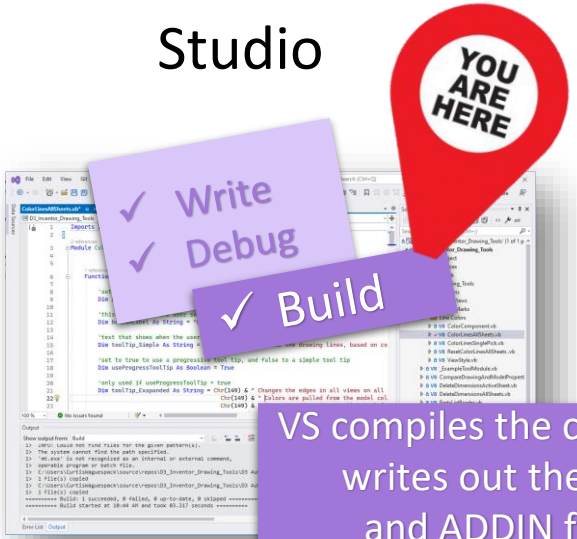
- If you wanted this to run an external ilogic rule, you could use this alternative

Overall Process

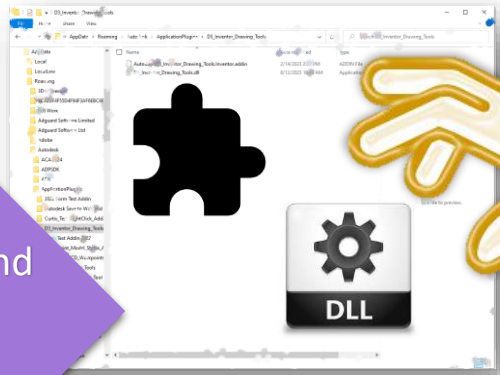
Visual Studio

Autodesk Application Plugins Folder

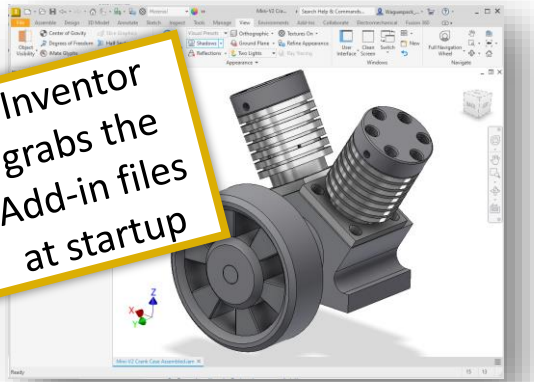
Autodesk Inventor



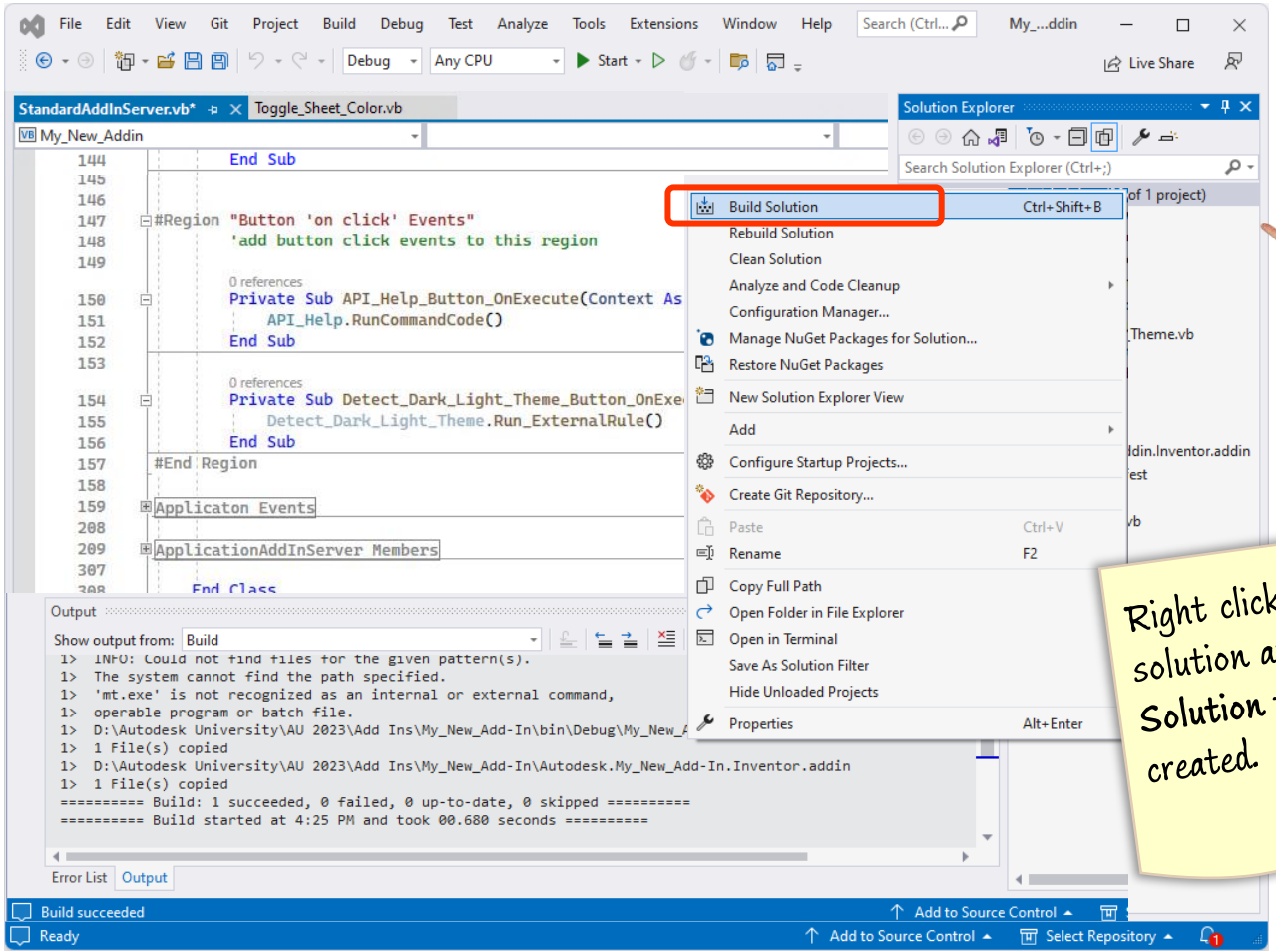
VS compiles the code and writes out the DLL and ADDIN files



Inventor grabs the Add-in files at startup



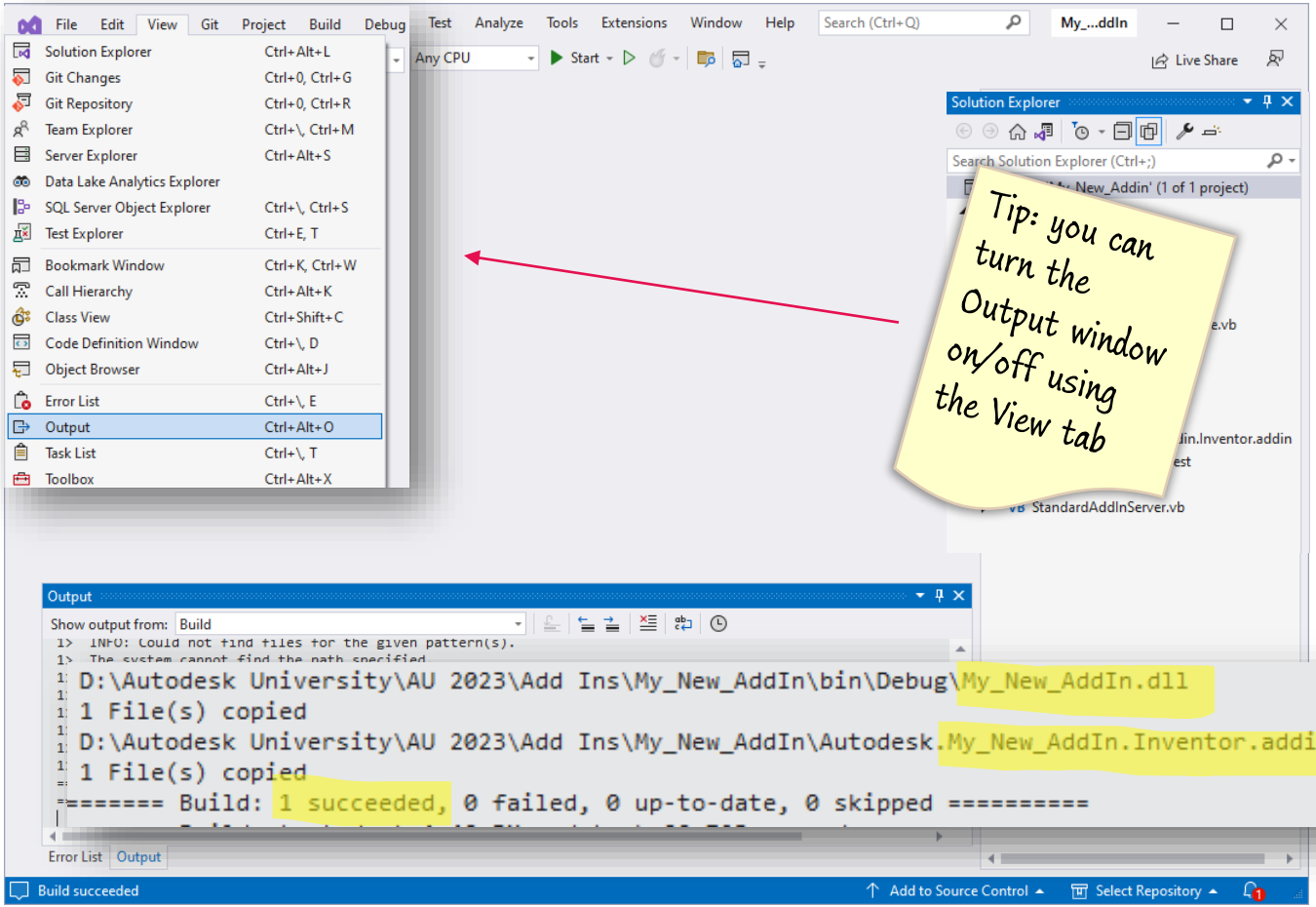
Compiling the Add-in



- The solution is shown in the Solution Explorer

Right click on the solution and choose **Build Solution** to see the files created.

Compiling the Add-in

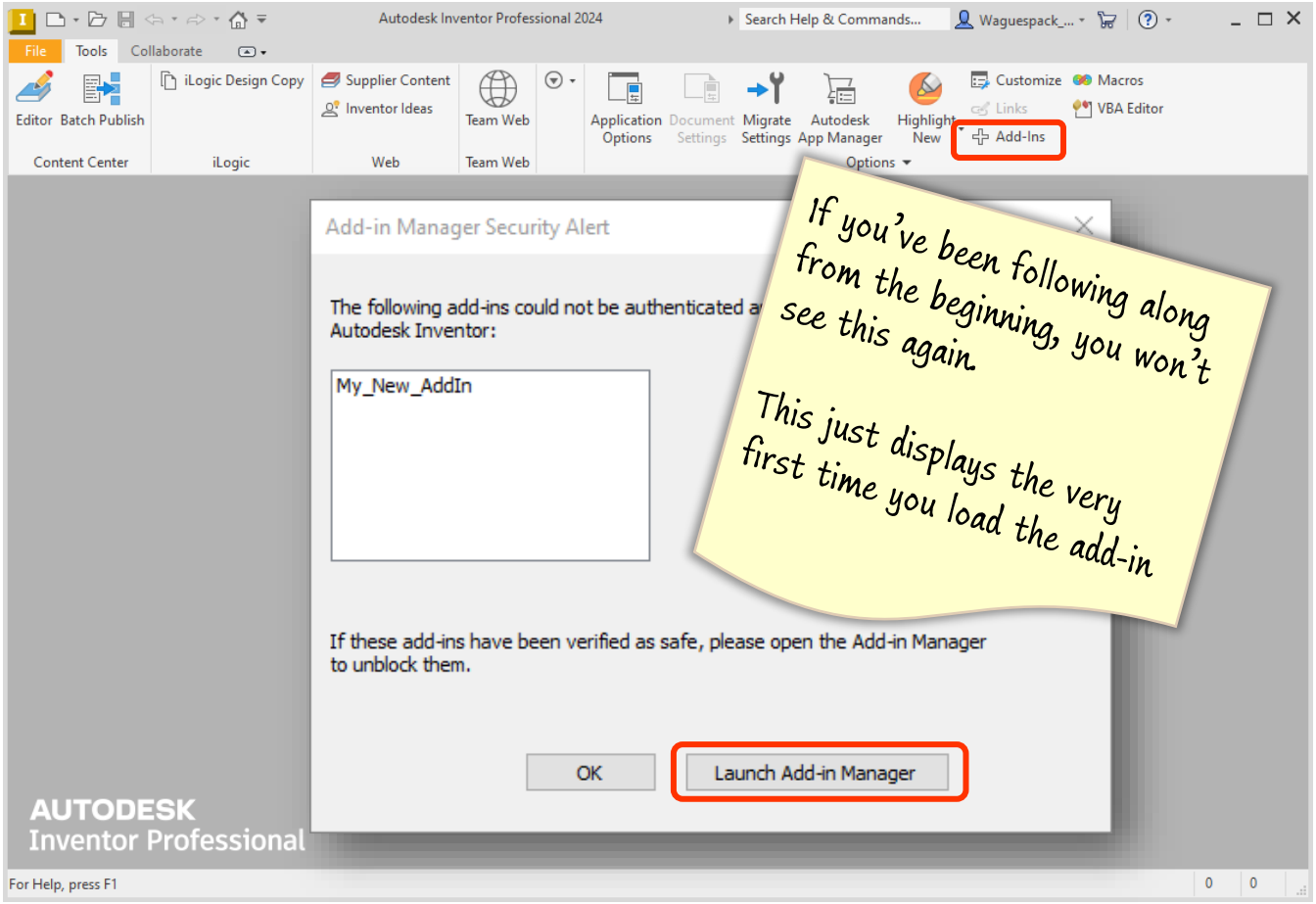


- The Build results are shown in the Output window

Testing the add-in

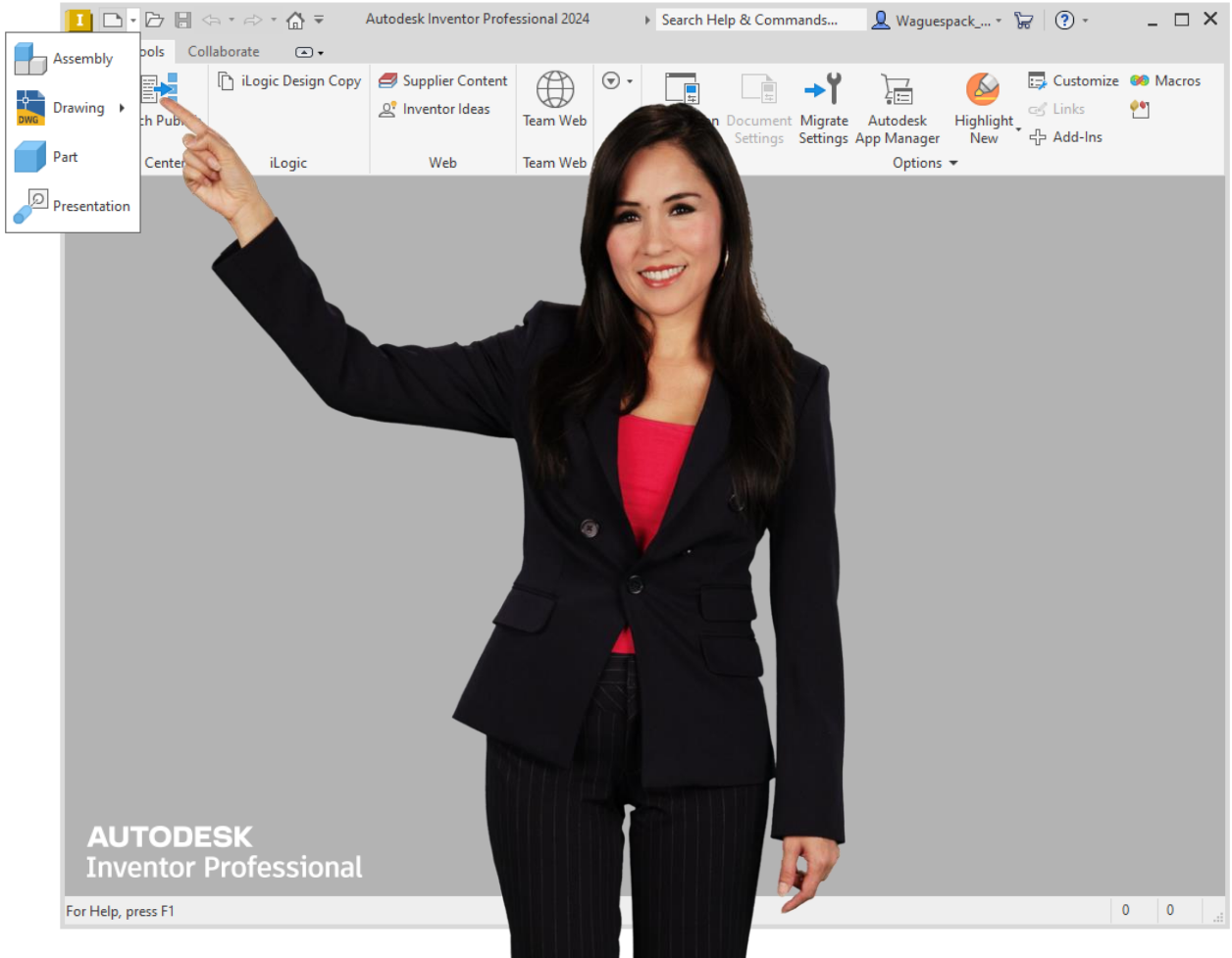
Working in Inventor

Testing the Add-in



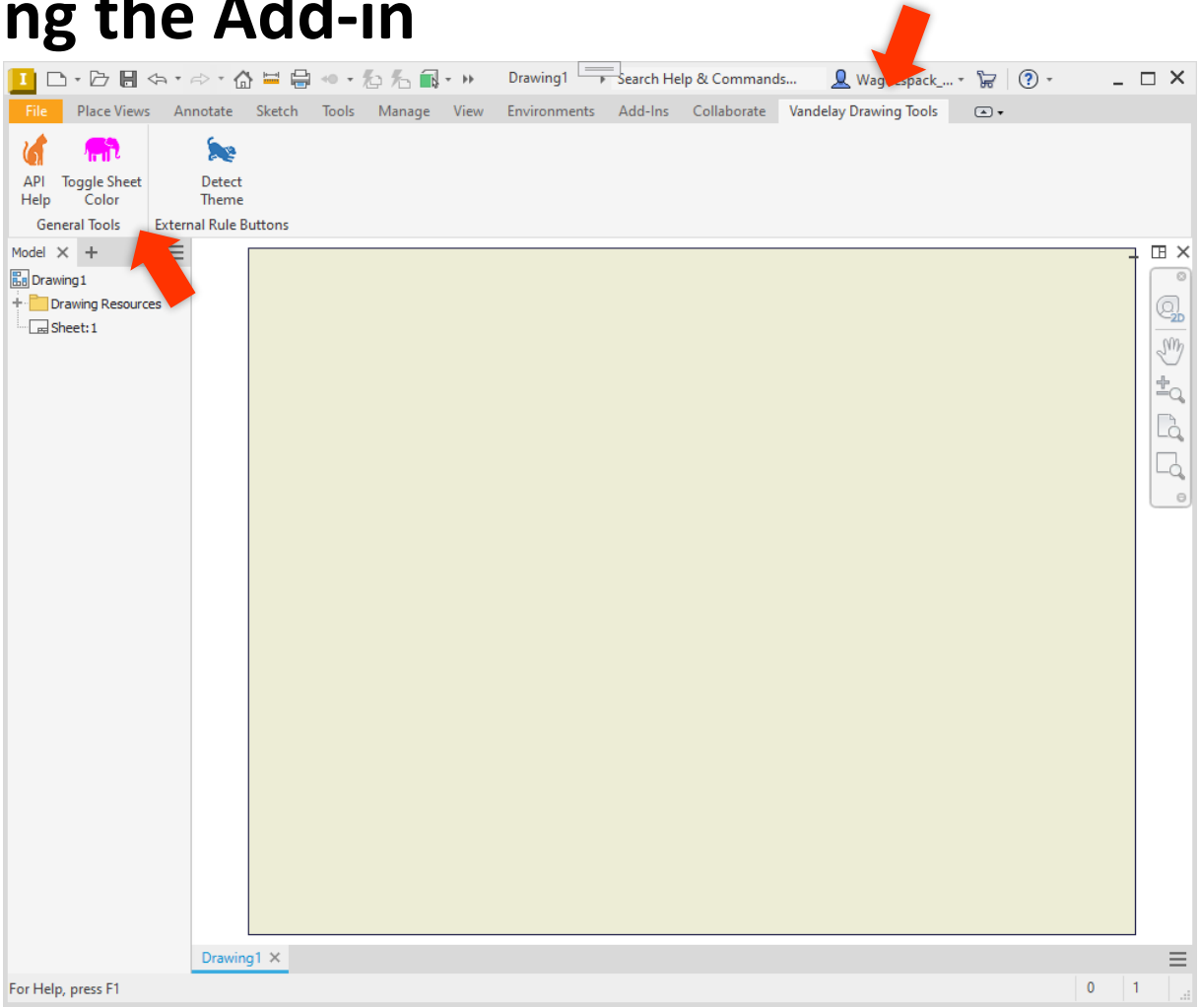
- Open Inventor.
- Note that you will be greeted by a message informing you that your add-in was blocked from loading
- Click the Launch Add-In Manager button or got to Tools > Add-ins

Using the Add-in



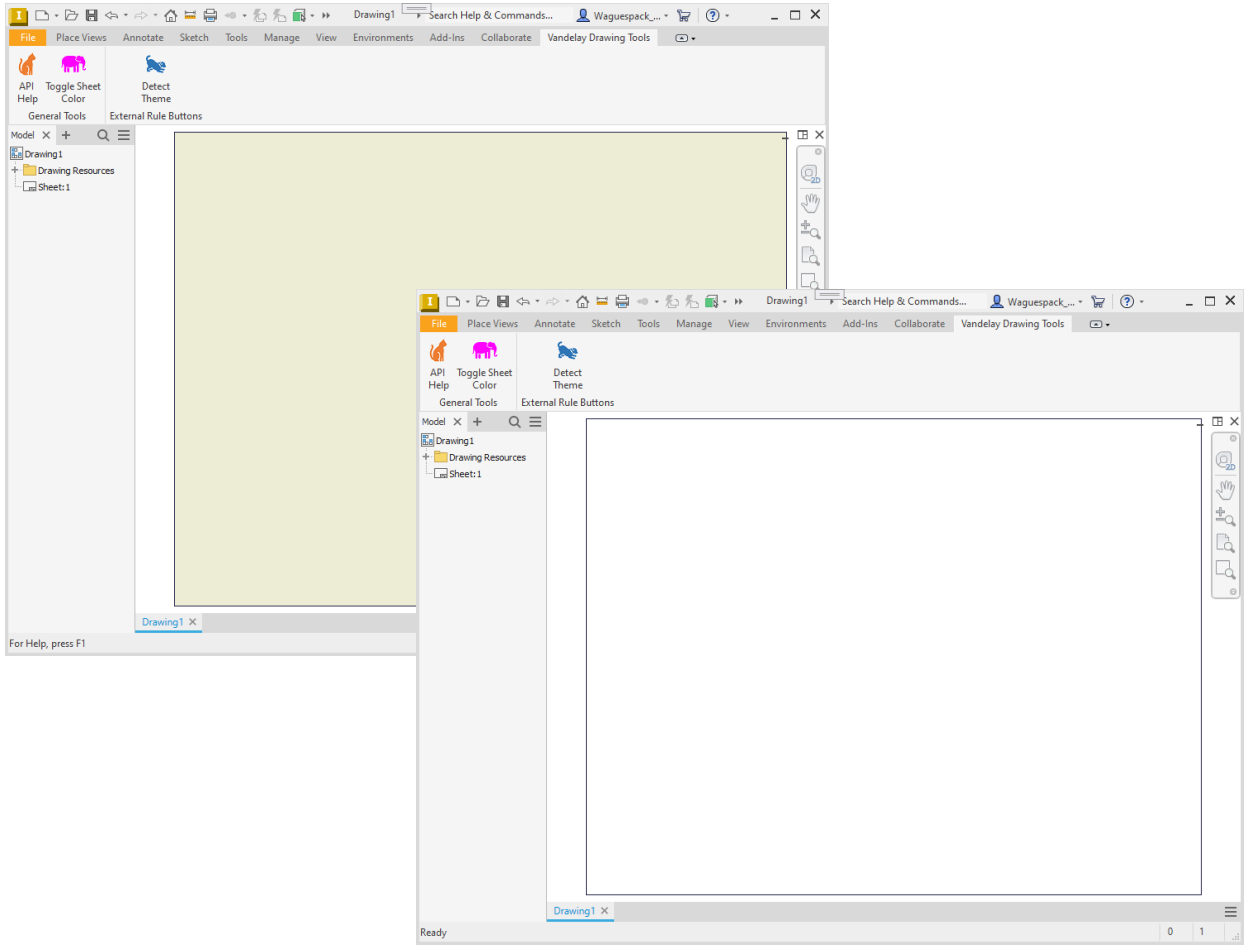
- Create a new drawing file from a template

Using the Add-in



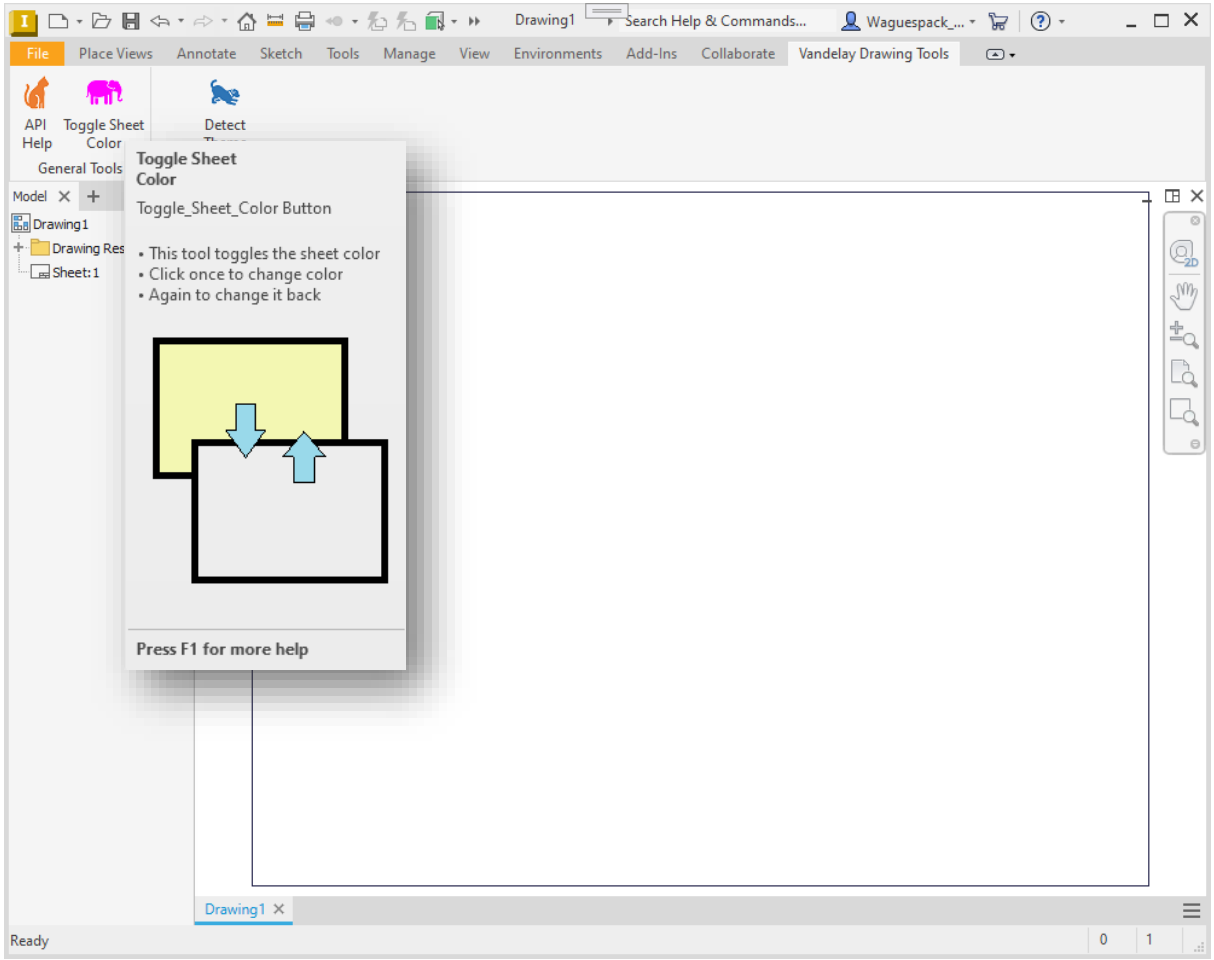
- Switch to the **Vandelay Drawing Tool** tab
- Note the new button on this tab

Using the Add-in



- When you click the button the code in the Toggle_Sheet_Color module will execute
- And the sheet color will toggle from the default color to white

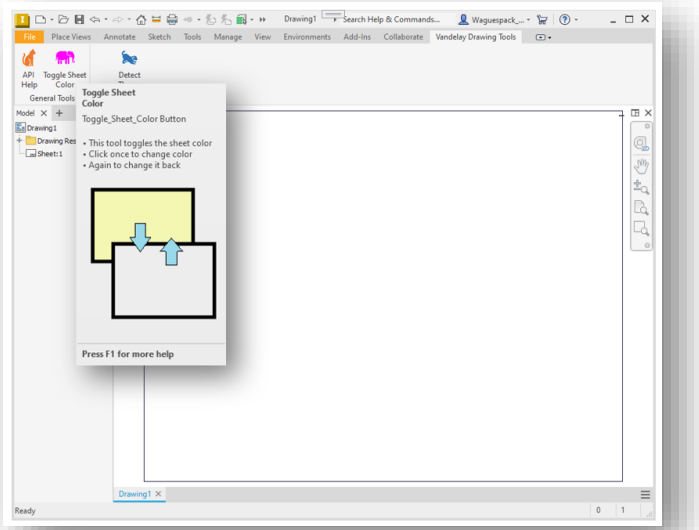
Using the Add-in



- Finally hover over the new button to see the expanded tooltip

Goal: be able to create new Inventor add-ins in just minutes!

You did it!



Quick Review

- Copy an existing Command Module to create a new one
- Open the module and change the module name
- In the CreateButton function:
  - Add button icons to the Resources
  - Set the button icons
  - Set the button label name
  - Set the simple tooltip
  - Set the progressive tooltip is desired
- In the Sub (where the code that does the real work resides)
  - Add the code
  - Alternatively add the line to run and external ilogic rule

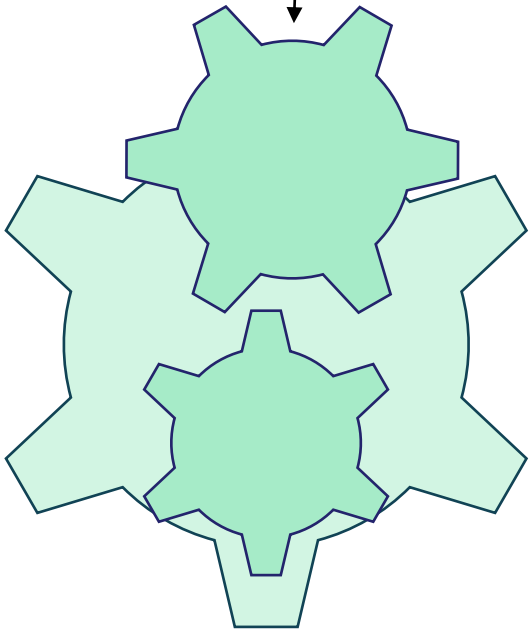
- In the StandardAddInServer
- Add the “WithEvents” line at the top
  - This creates the button definition name
- In the Create Buttons region:
  - Add a line to “wire up” the button definition name to the command module
- In the "Button 'on click' Events“ region:
  - Add an OnExecute sub to call the code in the module when the button is clicked

Review

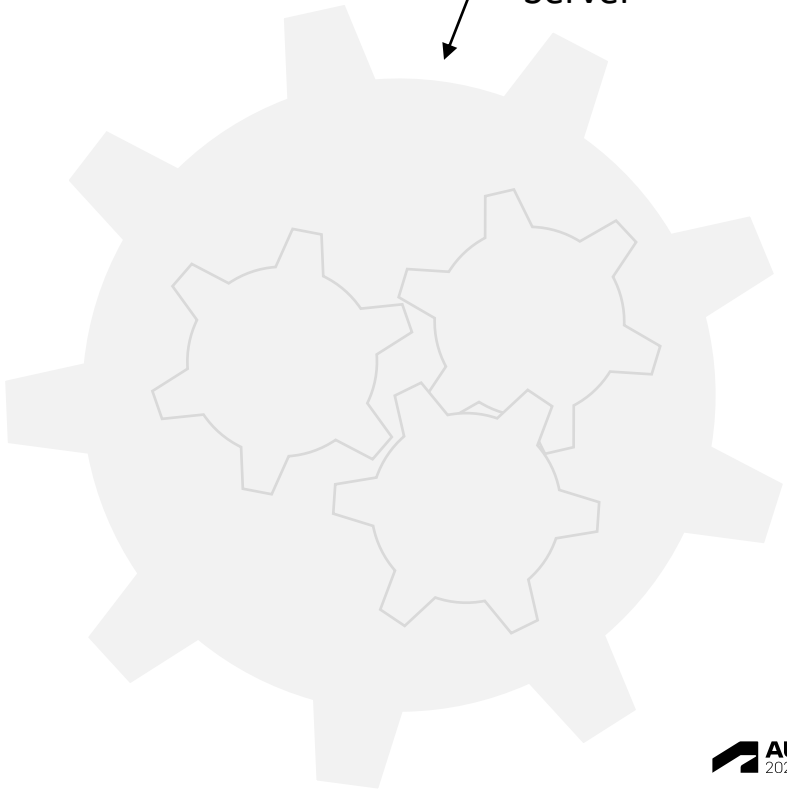
Create Button Utility

Picture Converter Utility

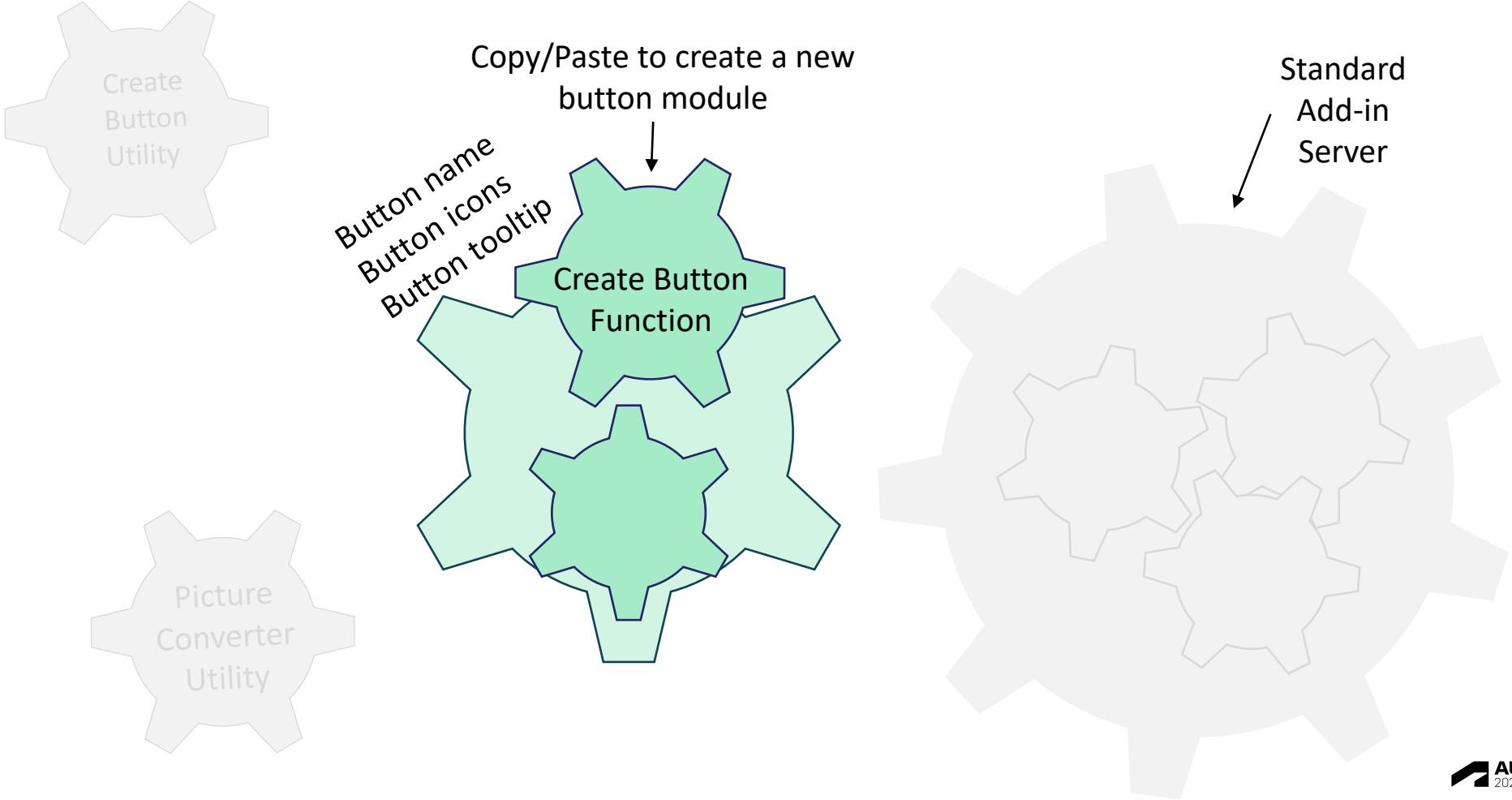
Copy/Paste to create a new button module



Standard Add-in Server

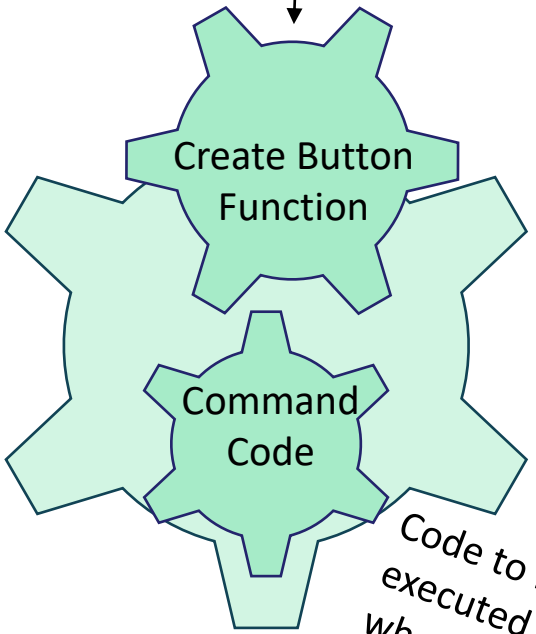


Review


Create Button Utility
Copy/Paste to create a new button module
Button name
Button icons
Button tooltip
Create Button Function
Standard Add-in Server
Picture Converter Utility

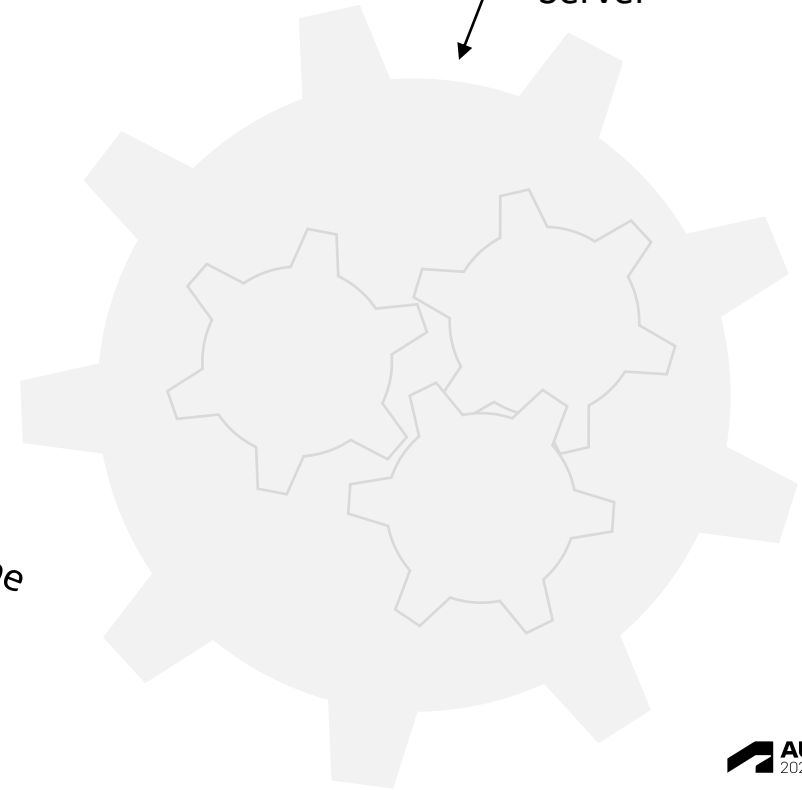
Review

Copy/Paste to create a new button module

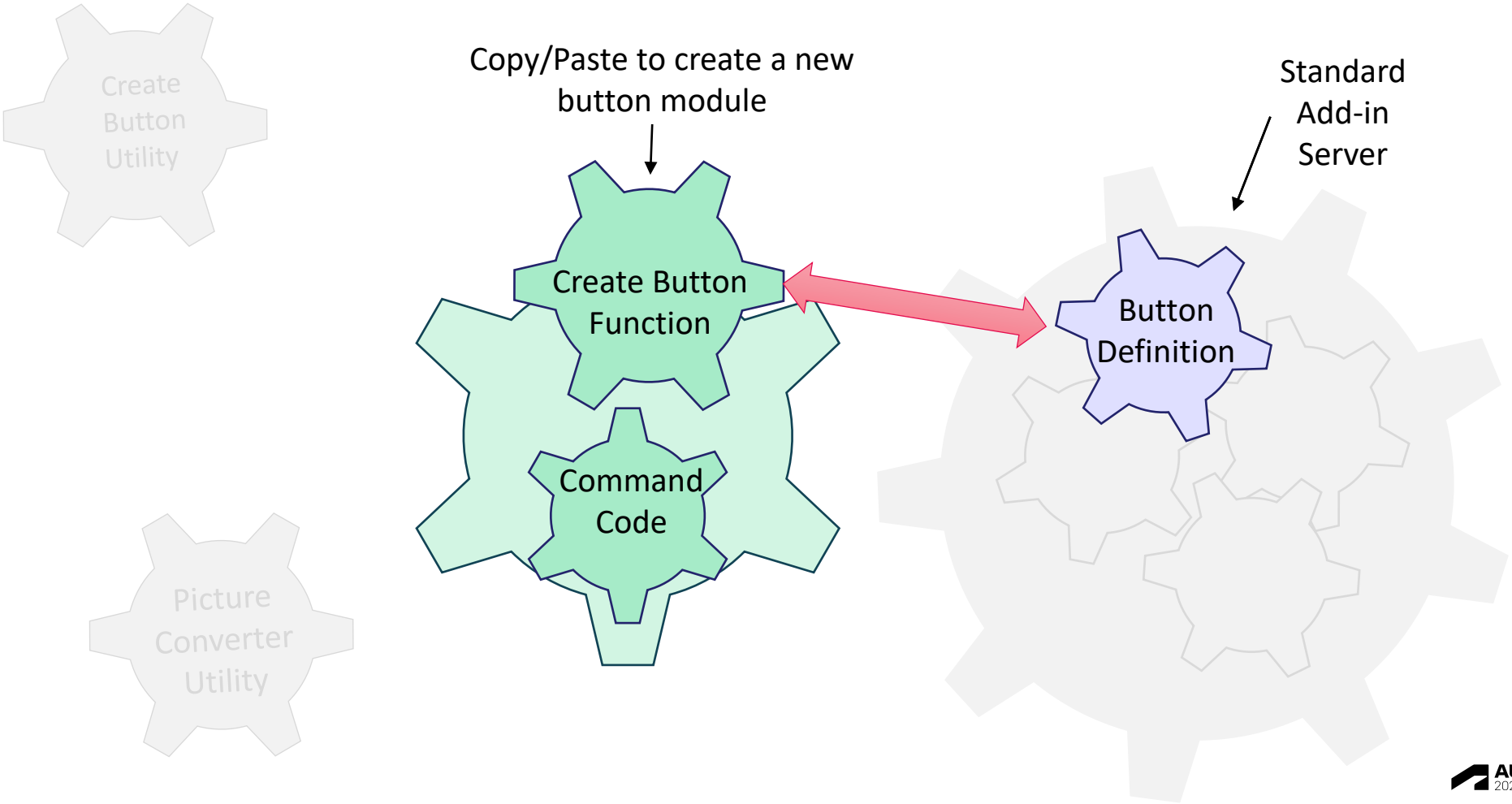


Code to be executed when the button is clicked

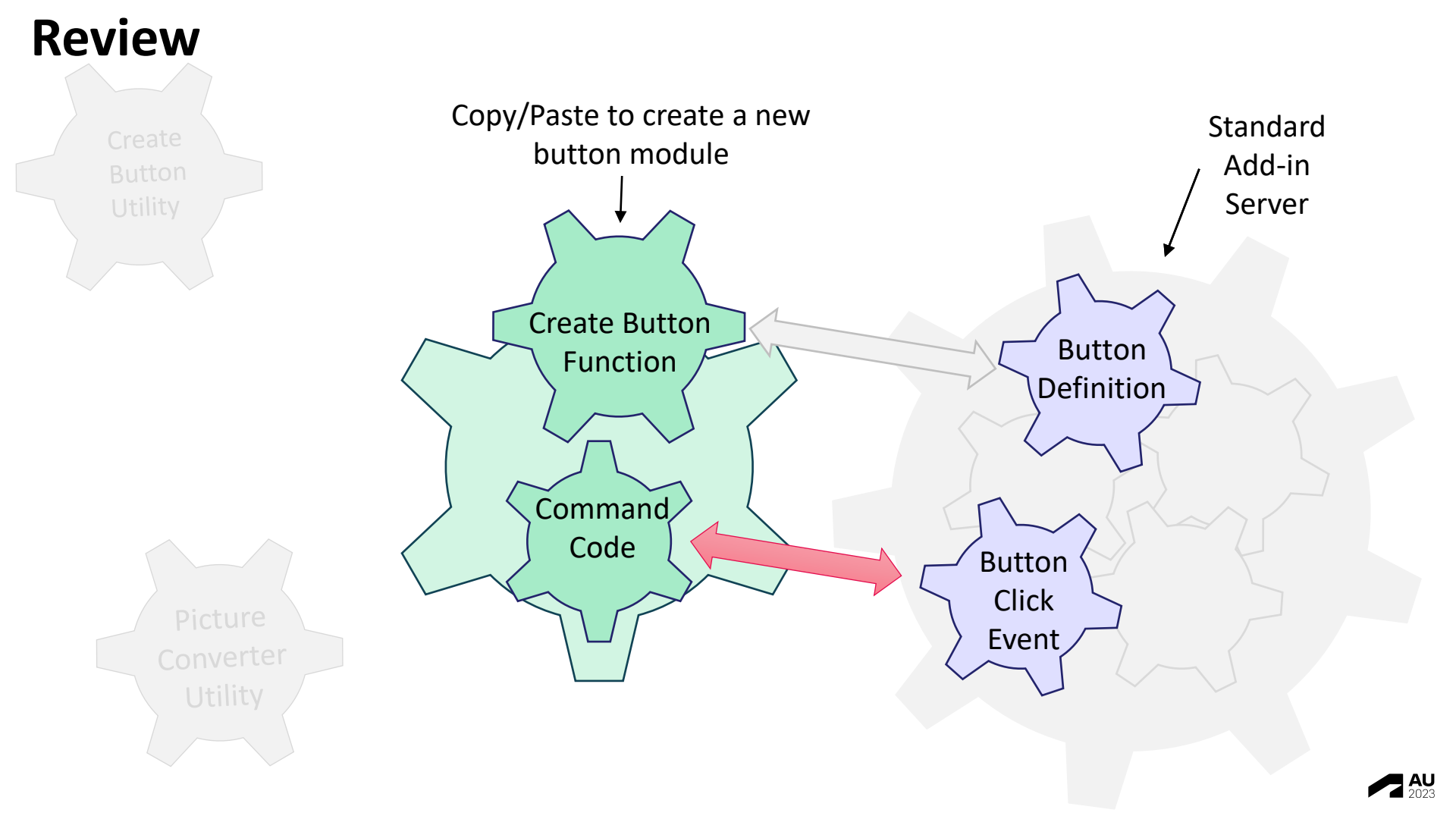
Standard Add-in Server



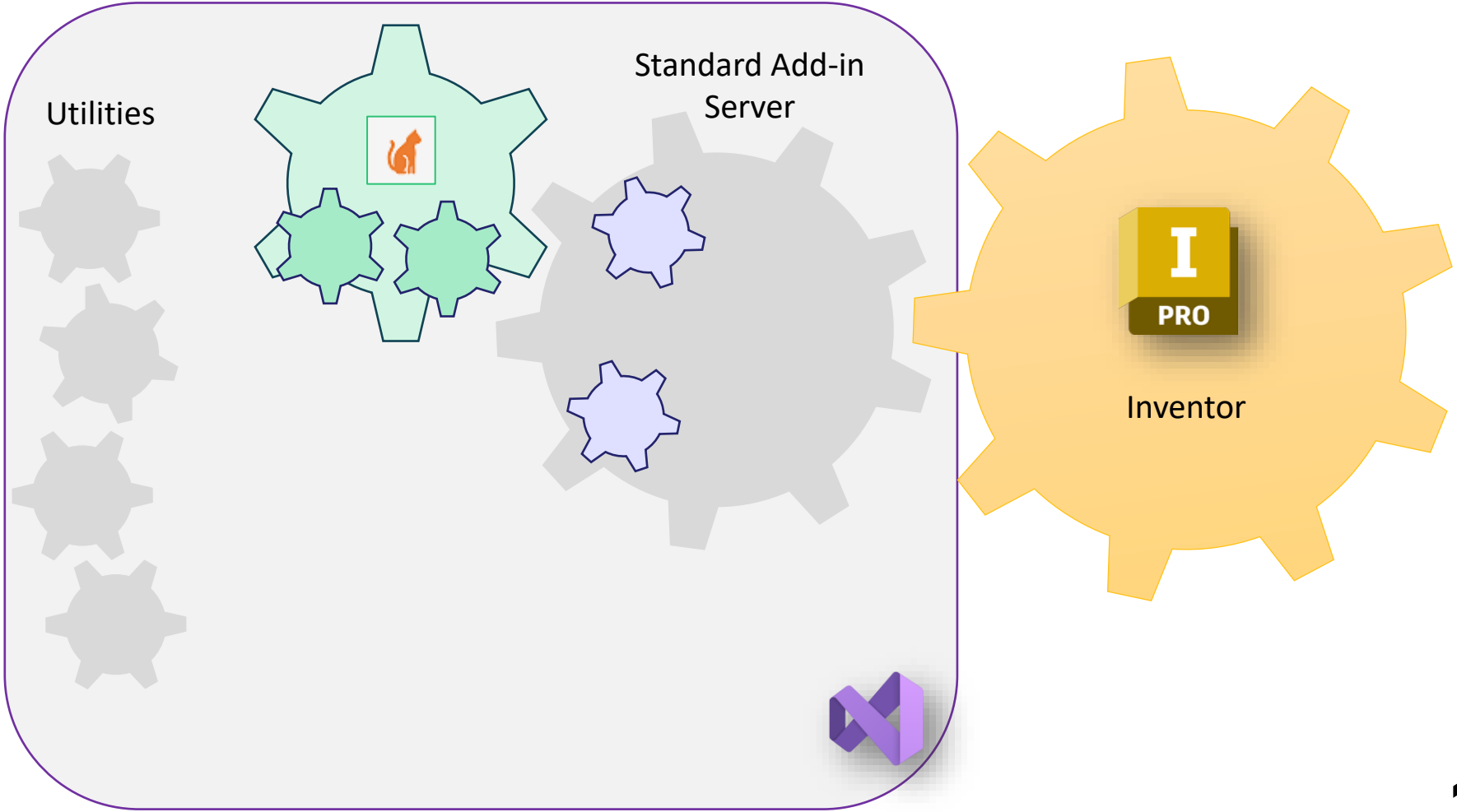
Review


Create Button Utility
Copy/Paste to create a new button module
Standard Add-in Server
Create Button Function
Button Definition
Command Code
Picture Converter Utility

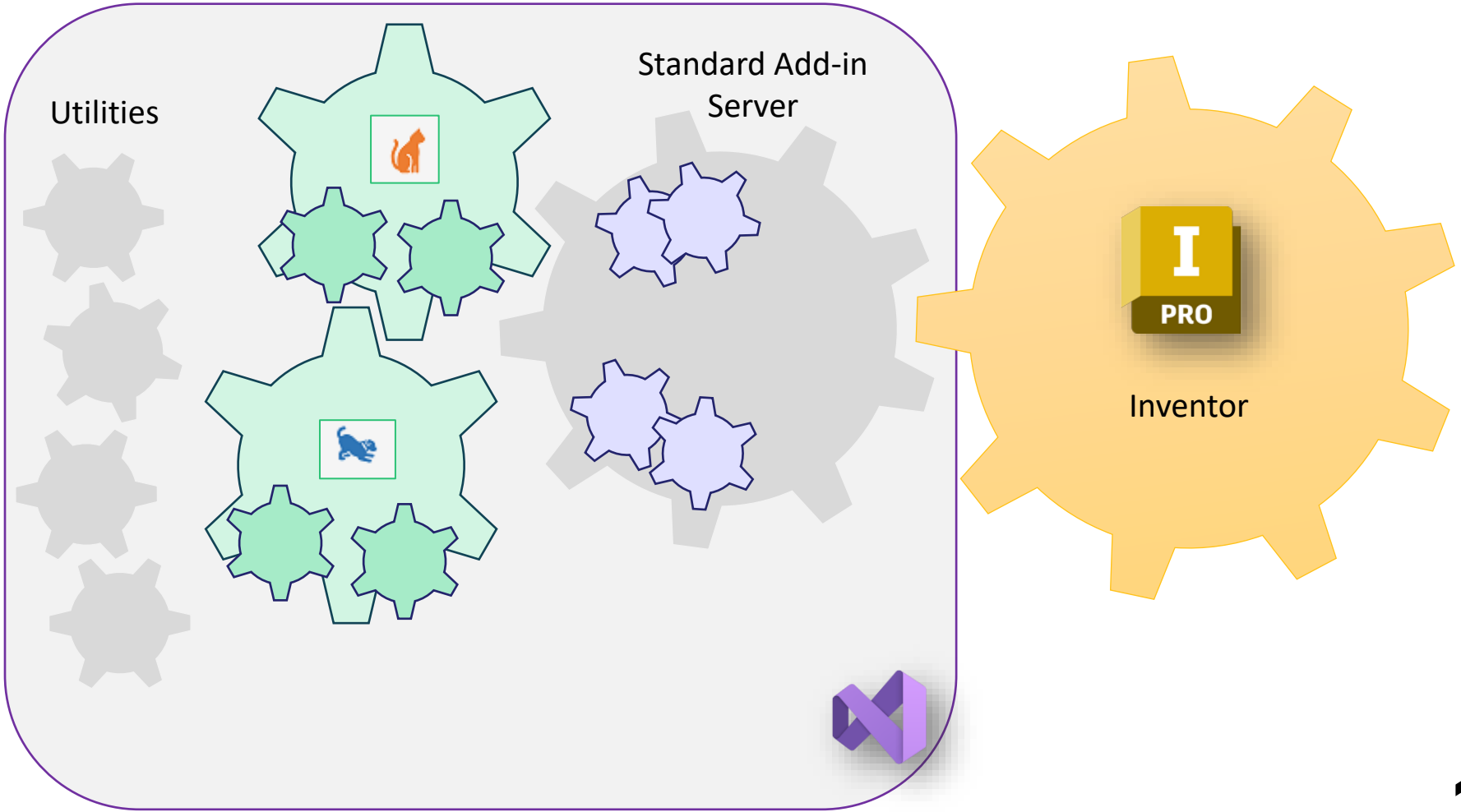
Review


Create Button Utility
Copy/Paste to create a new button module
Standard Add-in Server
Create Button Function
Command Code
Button Definition
Button Click Event
Picture Converter Utility

Review



Review



Understanding the Add-in Template Structure