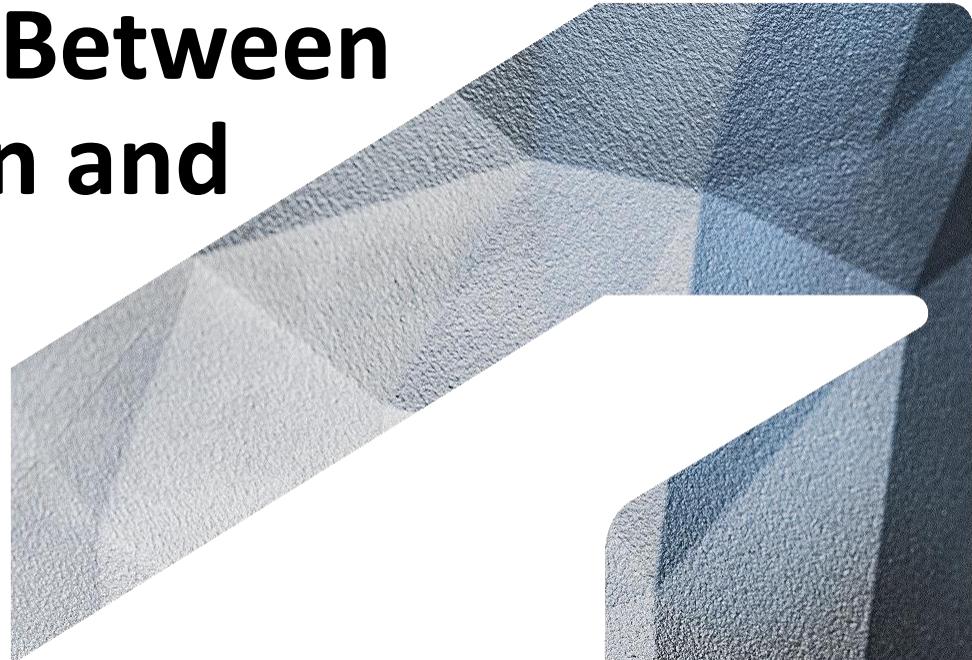


# Bridging the Gap Between iLogic Automation and Inventor Add-Ins

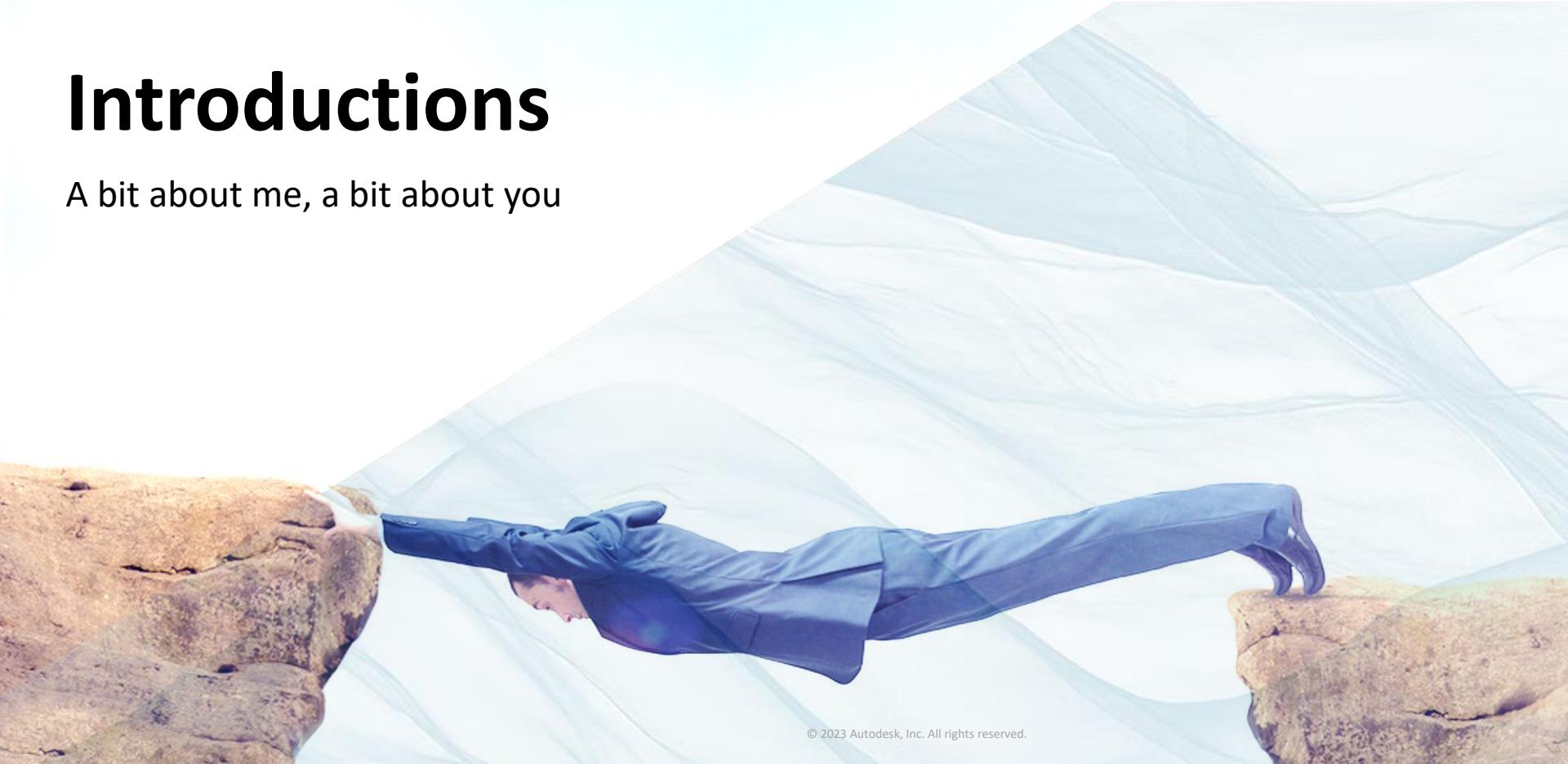
MFG601910

Curtis Waguespack  
Automation Solution Consultant Team D3



# Introductions

A bit about me, a bit about you



# About Me

Curtis Waguespack [wag] [əs] [pak]



Expert Elite Member



- I've used Inventor and AutoCAD for over 20 years, designing a number of manufactured products.
- This has included a great deal of Inventor automation with iLogic.
- I started out modifying LISP routines and writing VBA for AutoCAD.
- I currently work as an Automation Solution Consultant, where I help professionals like yourself automate their designs.
- I teach iLogic classes, and in the past, I have taught general Inventor and AutoCAD classes.
- Additionally, I've authored and co-authored multiple editions of the **Mastering Autodesk Inventor** book.



# About You

The Intended Audience

**This presentation is aimed at anyone who has:**

- existing iLogic rules in place
- run up against the limitations present within the automation tools offered with iLogic
- struggled to take things to the next level and create Inventor add-ins in the past
- concerns about having to start over with Inventor automation

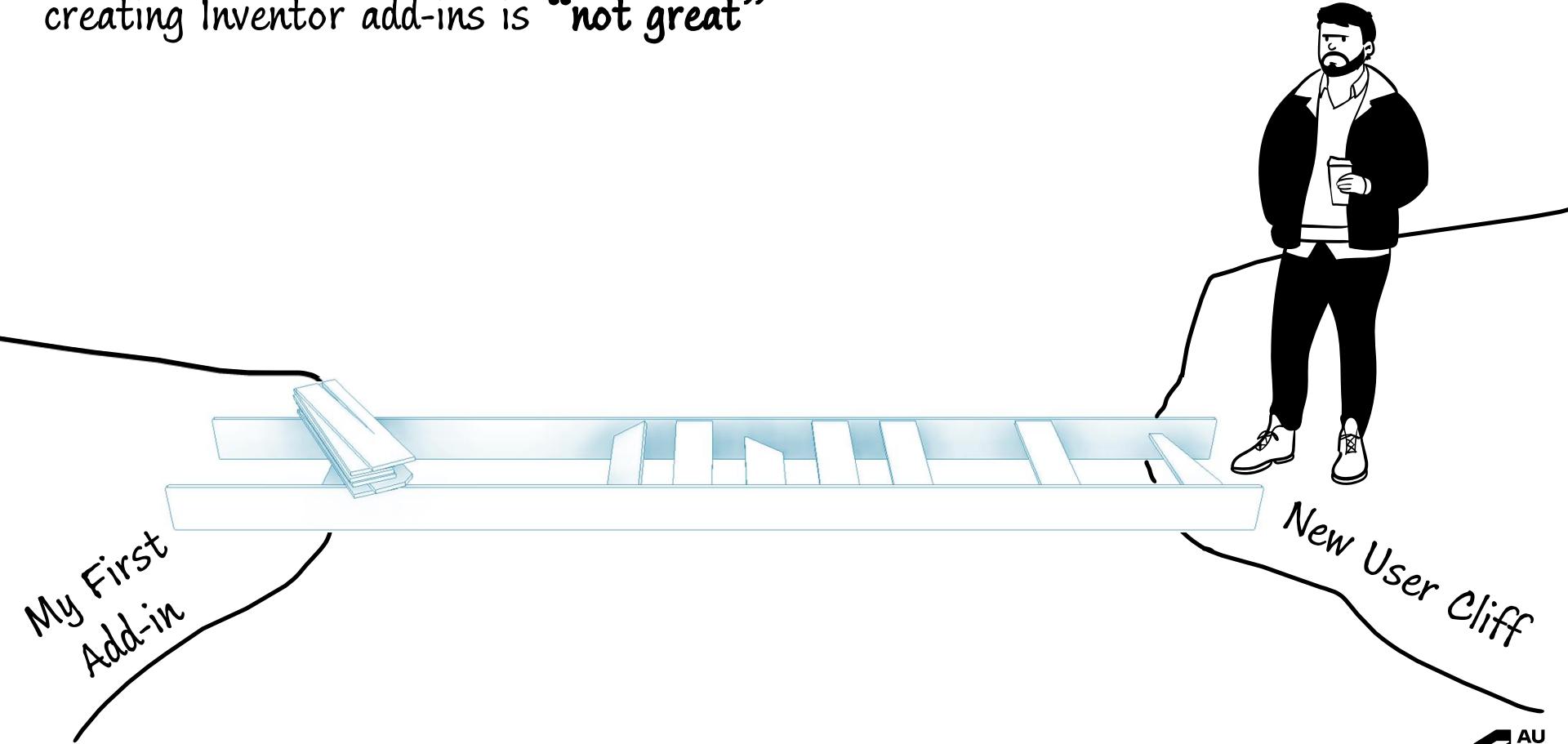


# About that bridge...

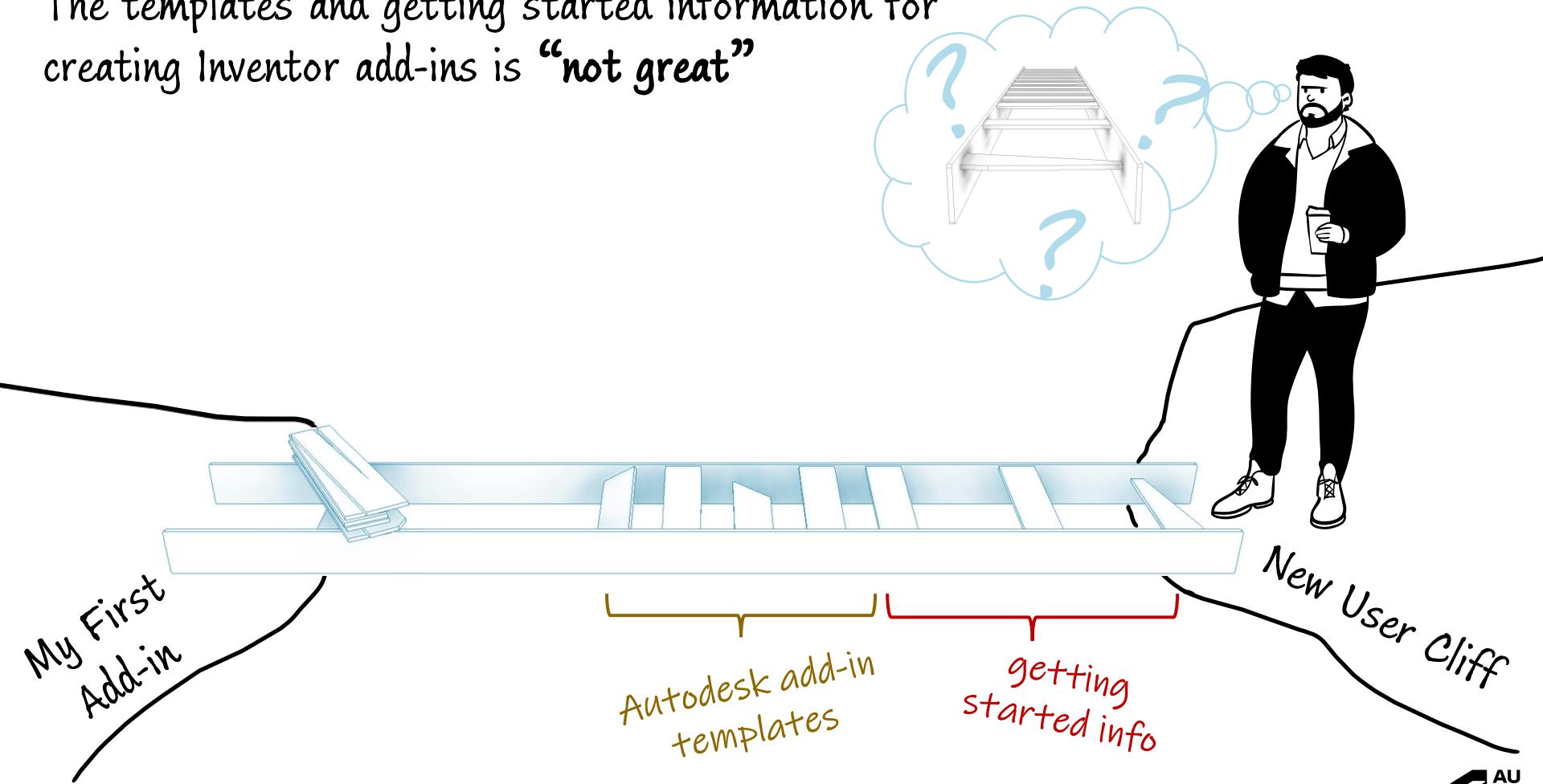
Understanding a new user's point of view



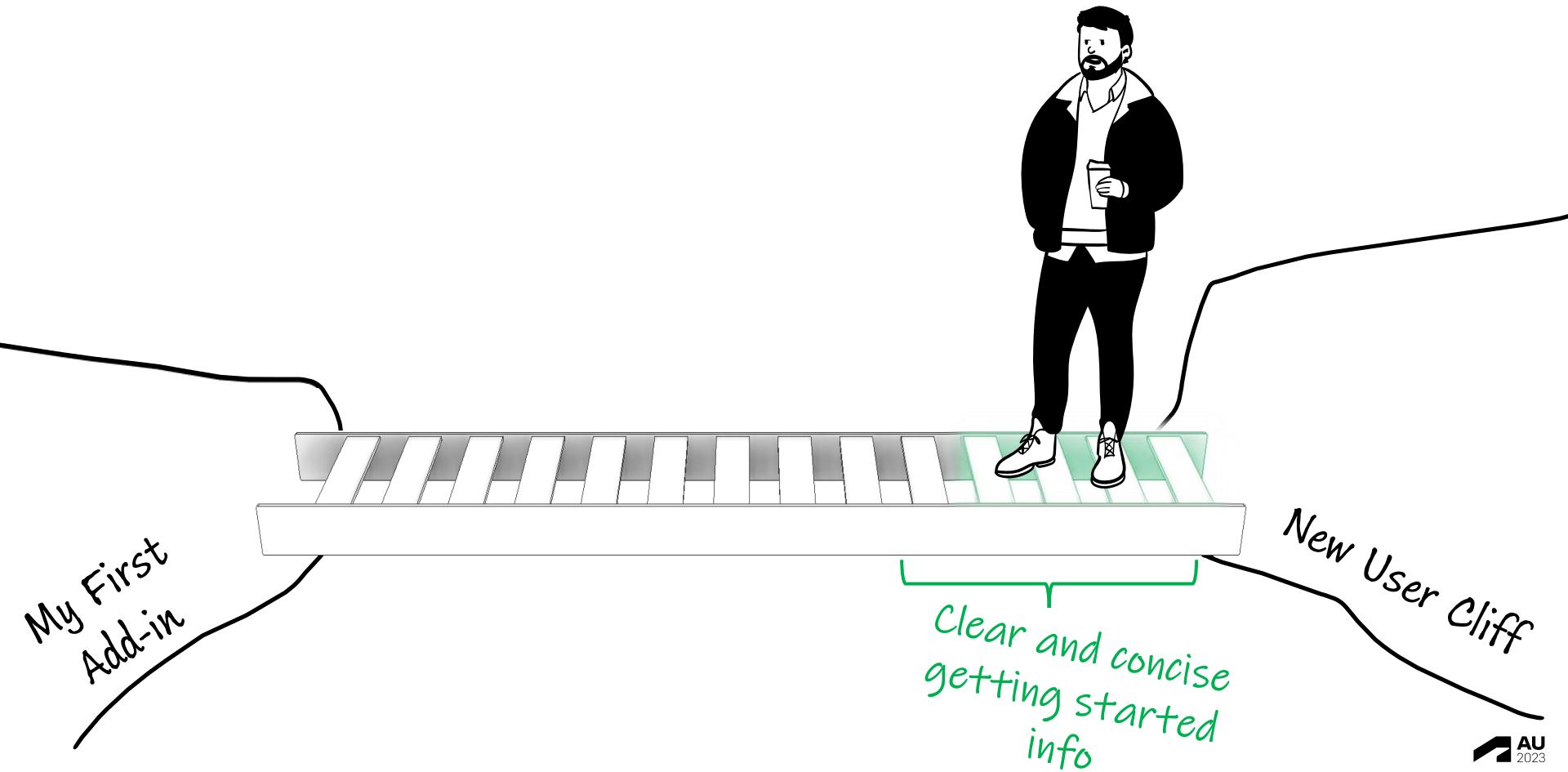
The templates and getting started information for creating Inventor add-ins is “not great”



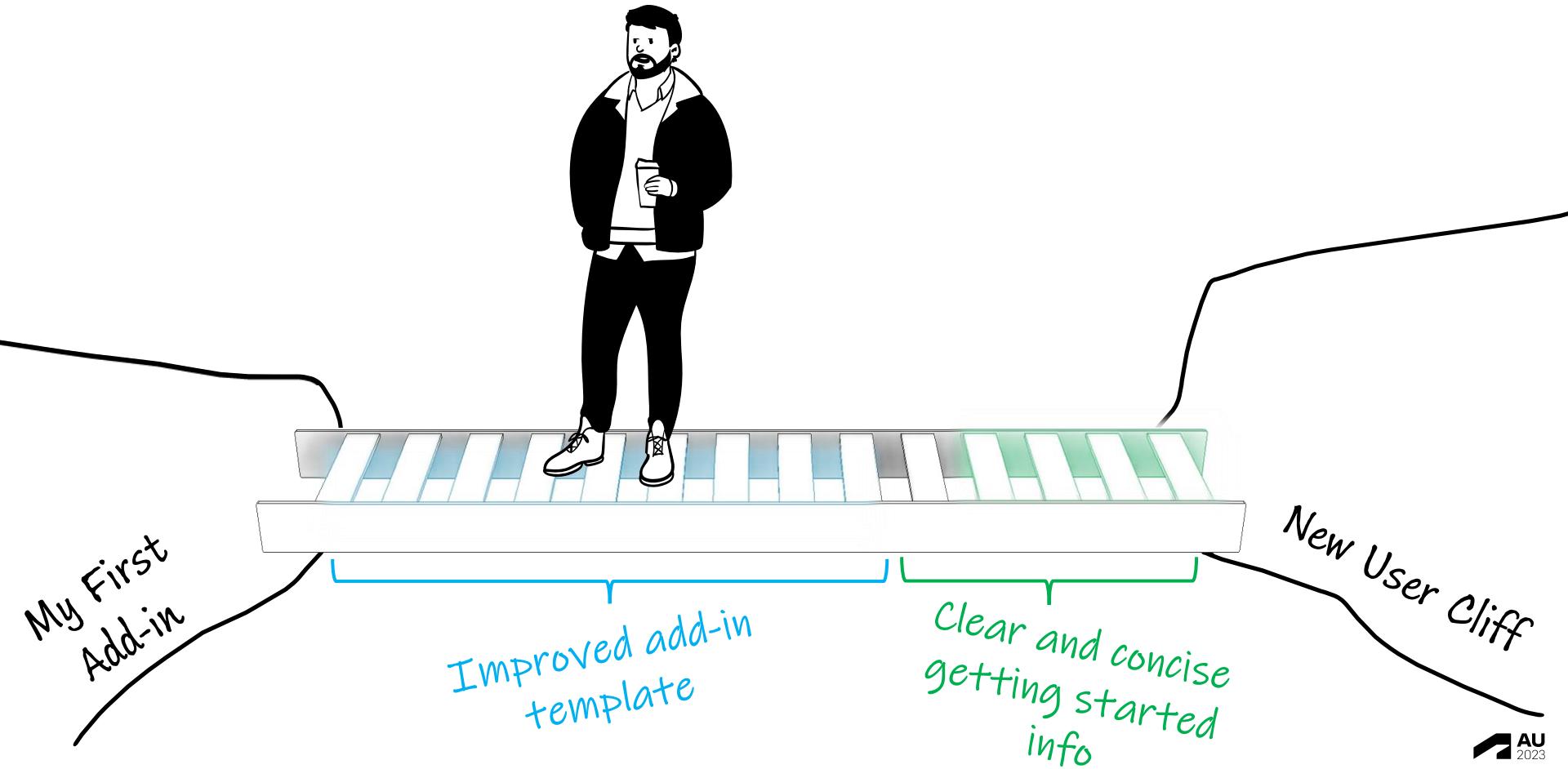
The templates and getting started information for creating Inventor add-ins is “not great”



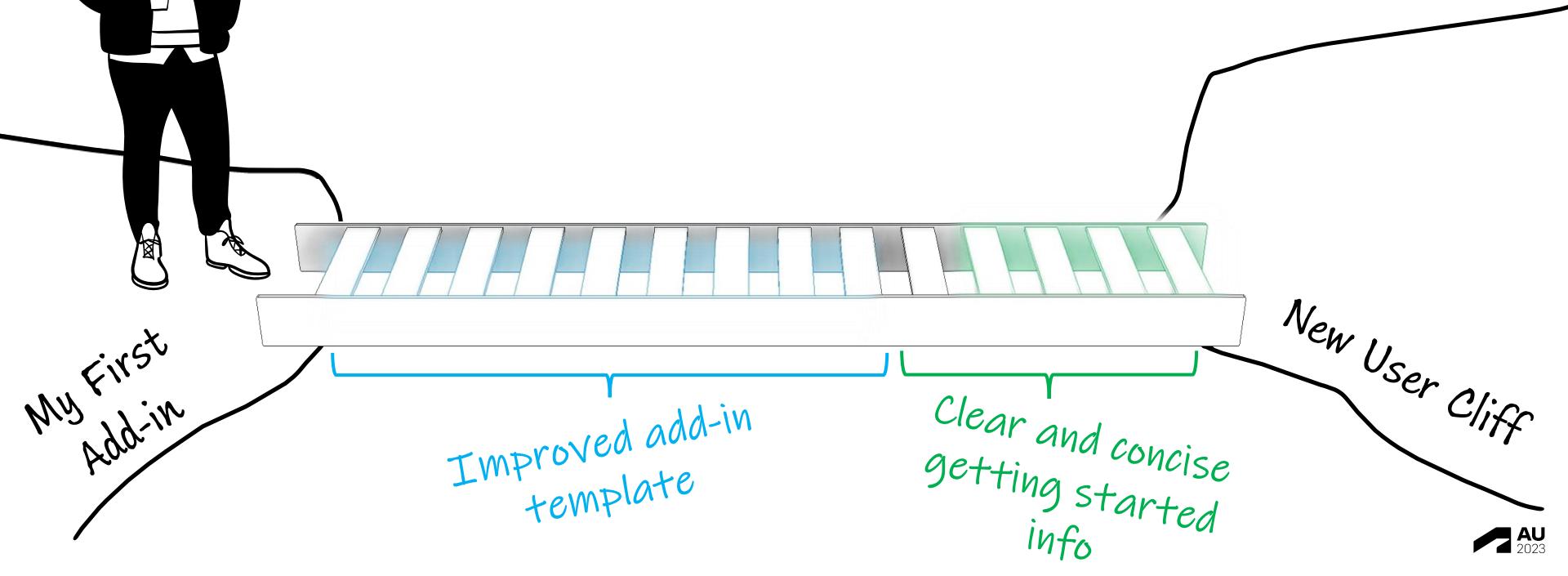
We're going to bridge this gap together.



# We're going to bridge this gap together.



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



# Why would we want to move to add-ins?

Why not just use iLogic?

# Motivations And Challenges



# Motivations

- Better Debugging Environment
- Would like to offer users a more organized & congruent interface & experience
- More control over events.  
Example: Trigger on the edit of a Parts List
- Would like to use C#



# Challenges

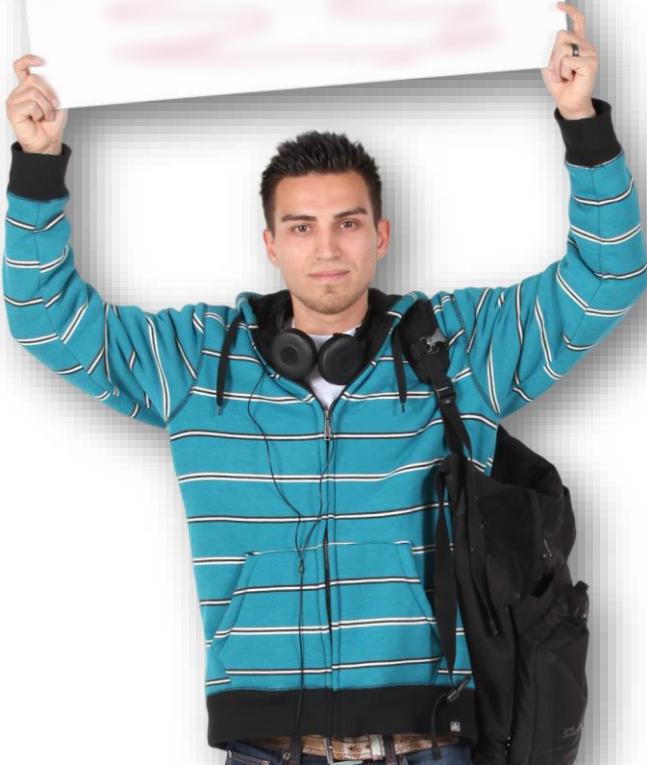
- Don't have the time to start over completely.
- Fall back on what we've been doing.
- Don't understand how to get started.
- Can't get the add-in templates to load.
- How do we use the add-in templates?



# Understanding Inventor Add-ins



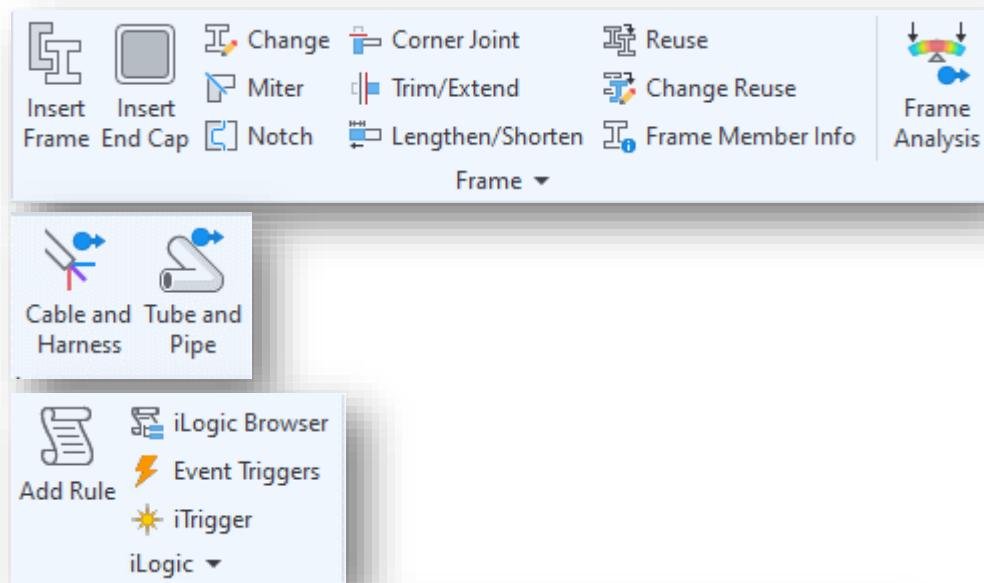
# What is an add-in?



- Also called a plug-in ( the 2 terms are interchangeable)
- Written and debugged in Visual Studio
  - Visual Studio is a stand-alone IDE application
    - IDE – Integrated Development Environment
- The Add-in is compiled to a DLL file and loaded into the Inventor process on startup of the Inventor application
- Add-ins are integrated into the Inventor's user interface
- Add-ins are integrated with all of Inventor's events
- Add-ins are typically written in VB.net or C#

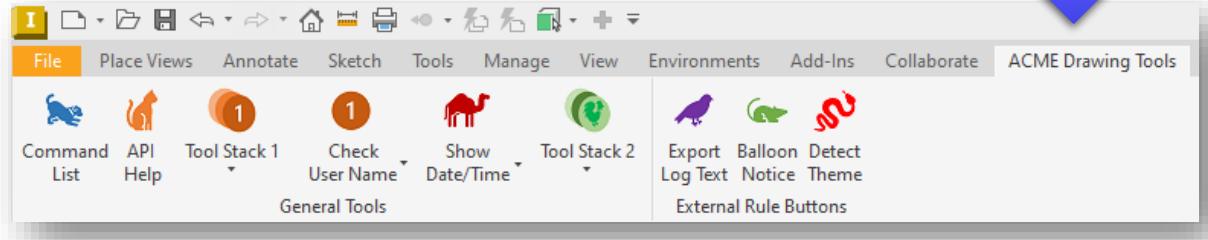
# Examples of Add-ins you might be familiar with

- Frame Generator
- Cable and Harness
- Tube and Pipe
- iLogic
- Inventor App Store downloads



 **AUTODESK** App Store

# Custom add-in

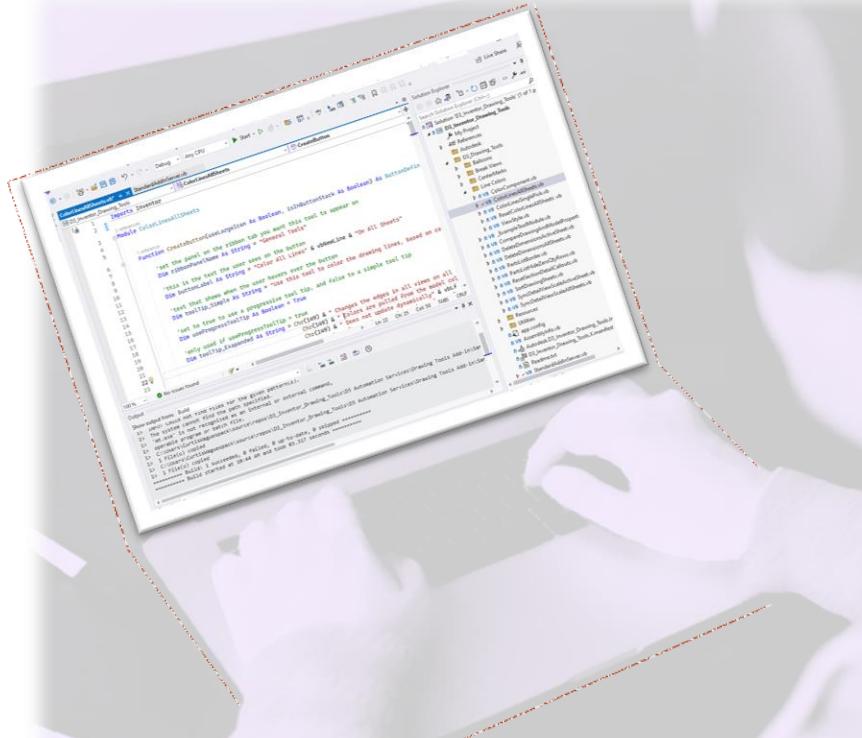


- A custom “enterprise” tab for your tools in each environment
- Buttons to run code from the add-in
- Buttons to run external iLogic rules

# How do we create an add-in?



# How do we create an add-in?

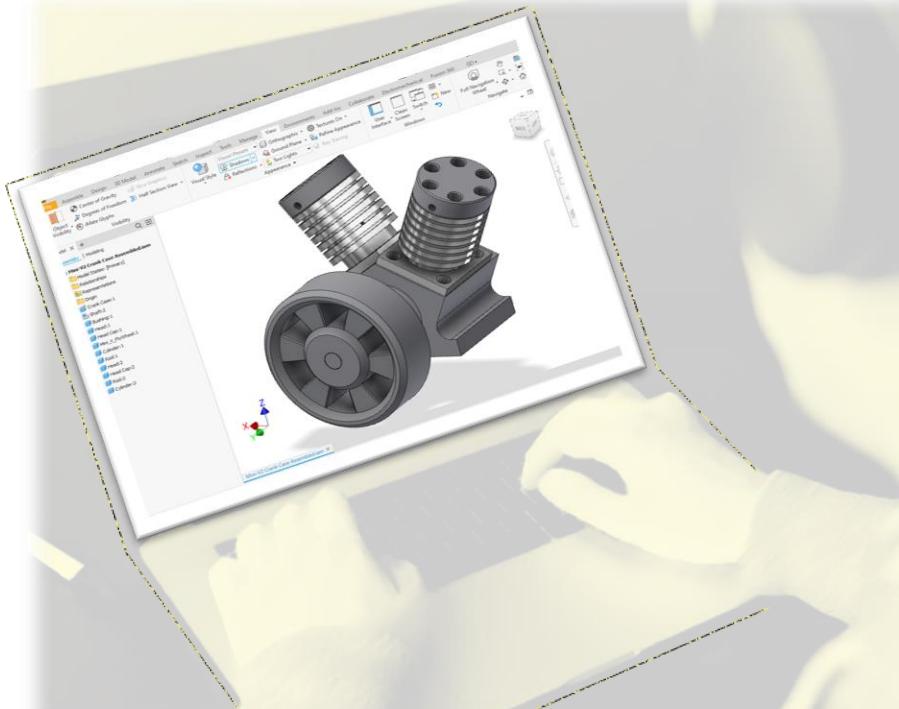


Writing and debugging the add-in

The Visual Studio logo, which consists of four interlocking purple and pink shapes forming a stylized 'M', is positioned at the top left. To its right, the words "Visual Studio" are written in a black sans-serif font. Below this, there is a vertical list of three items, each preceded by a blue square checkbox:

- Write Code
- Debug
- Compile

# How do we create an add-in?



**Loading and using the add-in**



Inventor

- Start Inventor
- Make sure the Add-in Loaded
- Use Add-in

# Writing the add-in

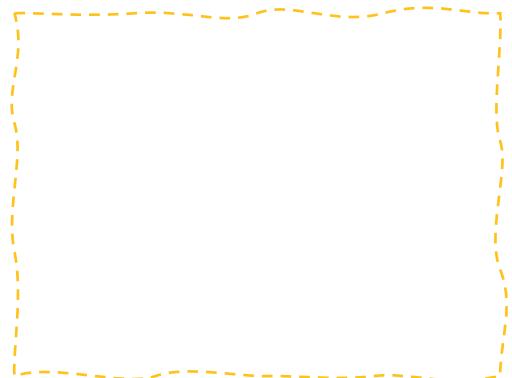


# Visual Studio



# Autodesk Application Plugins Folder

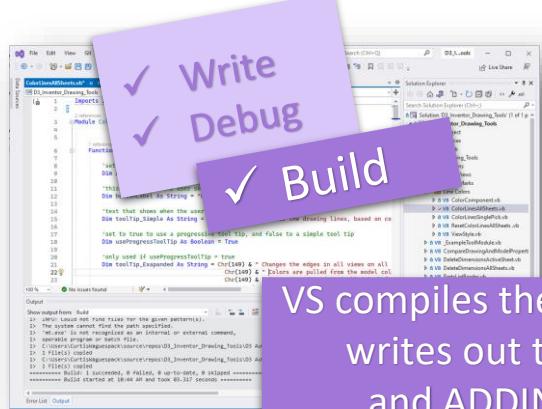
Autodesk  
Inventor



# The hand off



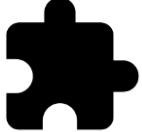
# Visual Studio



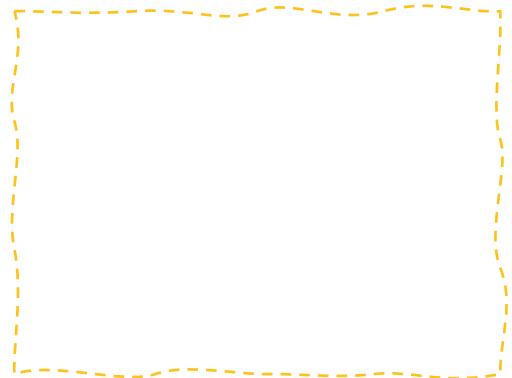
VS compiles the code and writes out the DLL and ADDIN files



# Autodesk Application Plugins Folder



# Autodesk Inventor



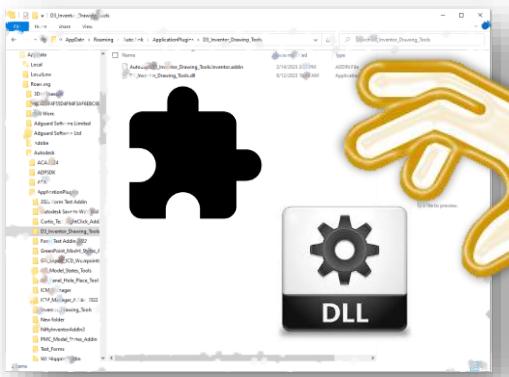
# Loading the add-in



# Visual Studio



# Autodesk Application Plugins Folder



A screenshot of the Autodesk Inventor interface. The main workspace shows a 3D model of a pump assembly with a fan-like impeller on the left and two cylindrical components on the right. A callout box with a yellow border and black text is overlaid on the left side of the screen, containing the text "Inventor grabs the Add-in files at startup". The Inventor ribbon menu is visible at the top, showing tabs like Home, Insert, Tools, View, and Manage. The Manage tab is currently selected. On the far right, there's a vertical toolbar with icons for various functions like Sketch, Draw, and Assembly. The status bar at the bottom indicates "Ready".



# Autodesk Inventor



# Overall Process



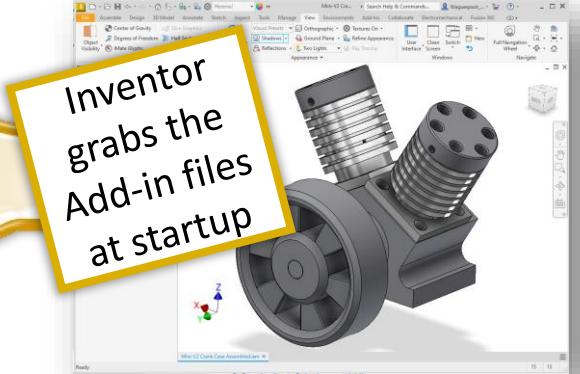
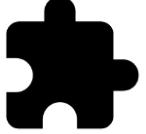
Visual  
Studio

```
    *Text that shows when the case
    *Set true to use a progressive tool tip, and false to a simple tool tip
    *Only used if userProgressToolTip = true
    *Set userProgressToolTip As Boolean = True
    *Changes the edges in all views to all
    *Changes the edges in all views to all
    *Pulled from the model col.
    *Pulled from the model col.
```

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



Autodesk  
Application Plugins  
Folder



Autodesk  
Inventor

# Getting Started

Installing Visual Studio and the Inventor  
SDK tools

\*SDK = Software Development Kit

# Steps to get started...

... with Visual Studio and the Inventor Software Development Kit

1



Install  
Visual  
Studio

2



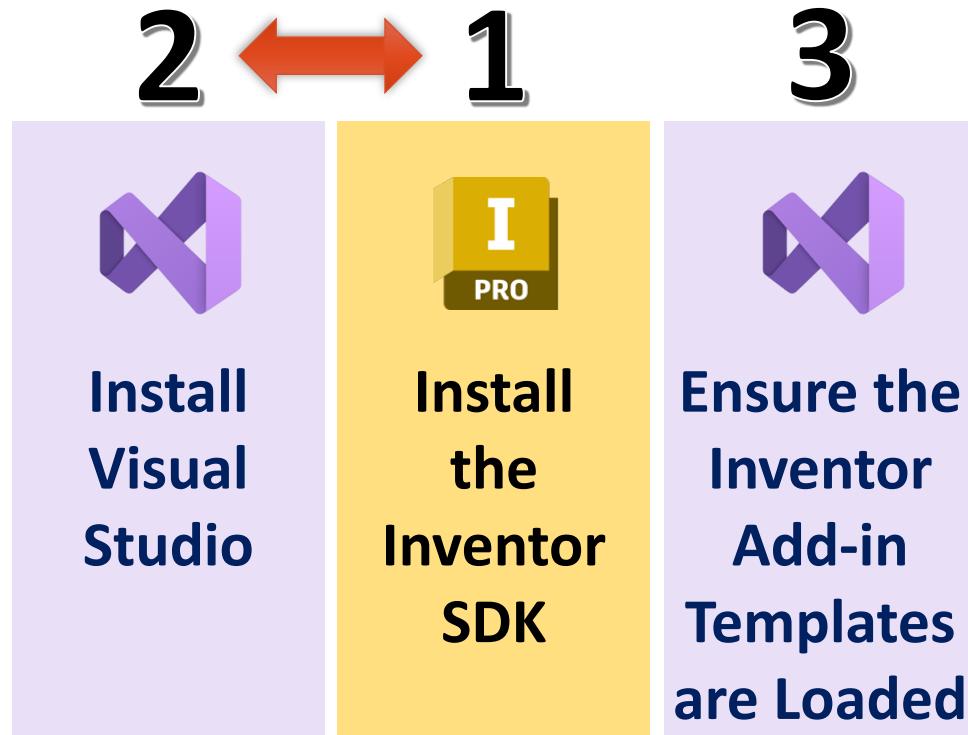
Install  
the  
Inventor  
SDK

3

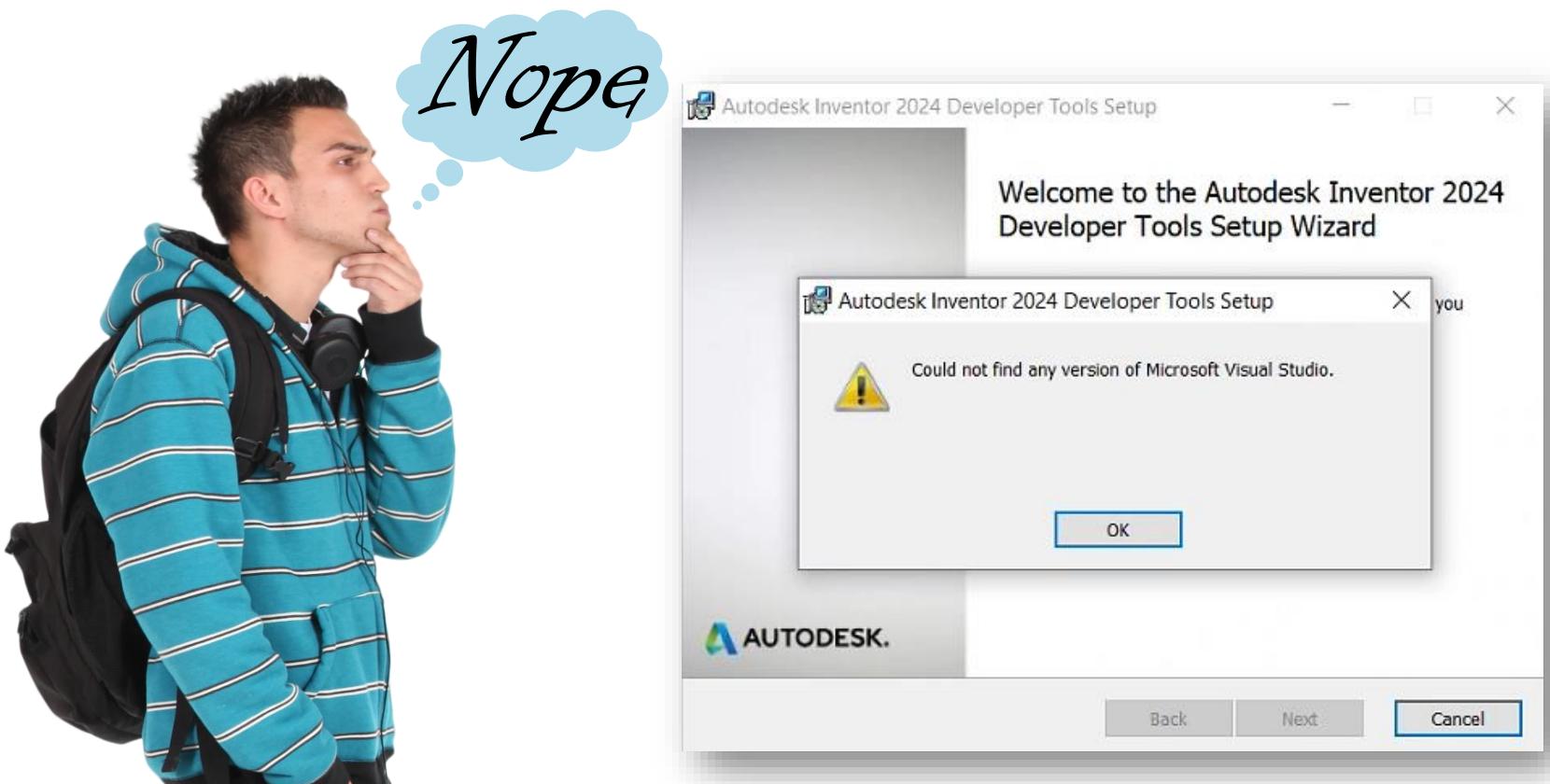


Ensure the  
Inventor  
Add-in  
Templates  
are Loaded

# Can we install the Inventor SDK before Visual Studio?



# Can we install the Inventor SDK before Visual Studio?



# Steps to get started...

... with Visual Studio and the Inventor SDK



- Install Visual Studio
- Extract the Inventor SDK ( software development kit)
- Ensure the Inventor Add-In Templates show up in Visual Studio

# Install Visual Studio

## Install Visual Studio

Article • 05/16/2023 • 20 contributors

Feedback

### In this article

- Step 1 - Make sure your computer is ready for Visual Studio
- Step 2 - Determine which version and edition of Visual Studio to install
- Step 3 - Initiate the installation
- Step 4 - Choose workloads
- Step 5 - Choose individual components (optional)
- Step 6 - Install language packs (optional)
- Step 7 - Select the installation location (optional)
- Step 8 - Start developing

Support or troubleshooting

See also

Show less

Applies to:  Visual Studio  Visual Studio for Mac  Visual Studio Code

Code

Welcome to Visual Studio 2022! In this version, it's easy to choose and install just the features you need.

<https://learn.microsoft.com/en-us/visualstudio/ide/?view=vs-2022>



The collage consists of five overlapping Microsoft Learn article cards:

- What is Visual Studio?** Article • 05/05/2023 • 12 contributors. In this article: Why use Visual Studio? Discover Visual Studio edition. Choose your Visual Studio edition. Install Visual Studio.
- Learn to use the code editor** Article • 03/08/2023 • 10 contributors. In this article: Create a new code file. Use code completion.
- Features of Visual Studio** Article • 03/08/2023 • 16 contributors. In this article: Modular installation. Create cloud-enabled Azure apps. Create web apps. Build cross-platform apps and games. Show 6 more.
- Tour the Visual Studio IDE** Article • 03/17/2023 • 14 contributors. In this article: Download and install. Start window. Create your project. Build your app. Show 4 more.
- Modular installation** This article describes features for experience already familiar with Visual Studio. For a basic introduction to the Visual Studio integrated development environment (IDE), we'll take a tour of some of the windows, menus, and other UI features. To develop any type of app or learn a language, you'll be working in the Visual Studio Integrated Development Environment (IDE). Beyond code editing, Visual Studio IDE brings together graphical designers, compilers, code completion tools, source control, extensions and many more features in one place.

# Which version of Visual Studio should I install?



Visual Studio Professional	Visual Studio Community	Visual Studio Express
<ul style="list-style-type: none"><li>• Full version</li><li>• Purchased from Microsoft</li><li>• Full Use Licensing</li></ul>	<ul style="list-style-type: none"><li>• Full Version</li><li>• Free from Microsoft</li><li>• Limited Use Licensing</li></ul>	<ul style="list-style-type: none"><li>• Limited version</li><li>• Free from Microsoft</li><li>• <b>Not suitable for Inventor add-ins</b></li></ul>

# Steps to get started...

... with Visual Studio and the Inventor SDK



- Install Visual Studio



- Extract the Inventor SDK ( software development kit)



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

# Steps to get started...

... with Visual Studio and the Inventor SDK



- Install Visual Studio



- Extract the Inventor SDK ( software development kit)



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





What is the  
Inventor  
SDK?

**SDK = Software Development Kit**



## **SDK = Software Development Kit**

The Inventor SDK includes samples and tools to demonstrate and describe the functionality that is available in the Inventor API\*

\* API = Application Programming Interface



## SDK = Software Development Kit

The Inventor SDK includes samples and tools to demonstrate and describe the functionality that is available in the Inventor API\*

\* API = Application Programming Interface

Wait... what  
is this API  
you speak of?

# Understanding the API

What is the  
Inventor  
SDK?



SDK = Software Development Kit

The Inventor S-

nd tools to  
onality that is available  
rogramming Interface

DETOUR



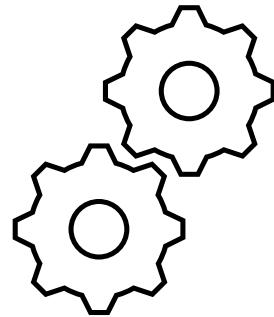
# Understanding what an API is (Web Page Example)



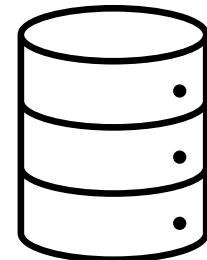
User  
Interface



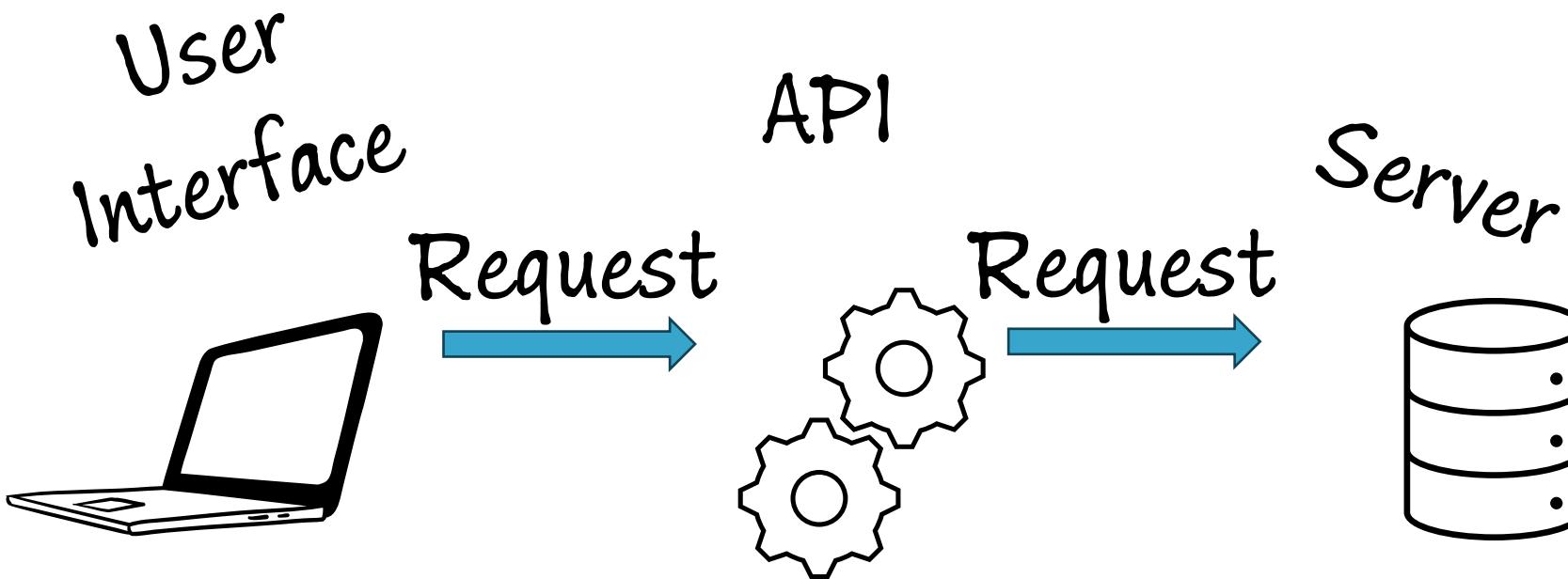
API



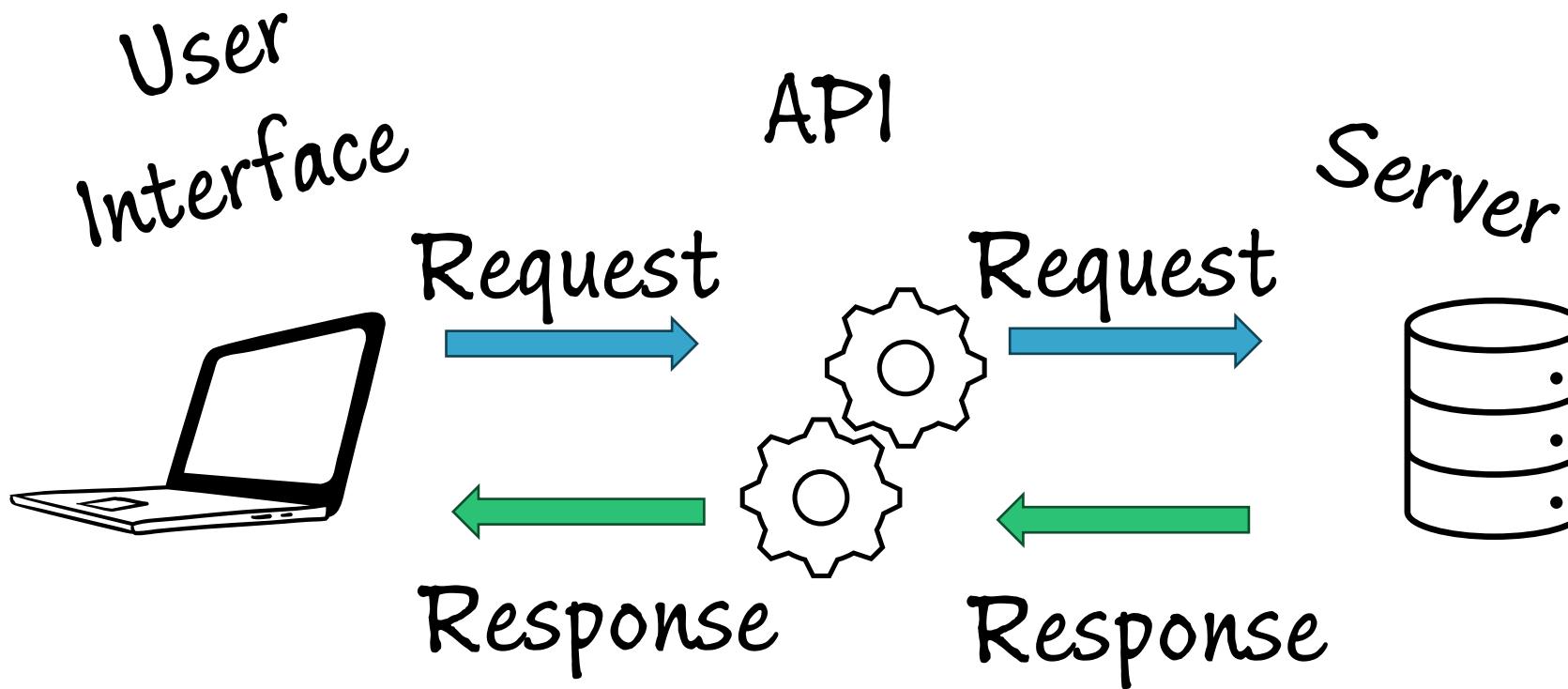
Server



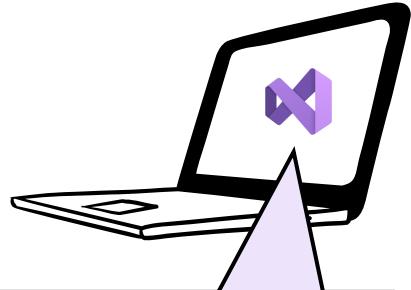
# Understanding what an API is (Web Page Example)



# Understanding what an API is (Web Page Example)



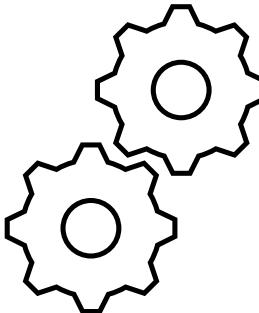
# Understanding what an API is (Inventor Add-in Example)



A screenshot of a code editor window showing a C# file with some test code. The code includes annotations for TestMethod and Assert.AreEqual, indicating it's a unit test for an API.

```
14 [TestMethod]
15 public async Task ReturnsNotFoundGivenInvalidId()
16 {
17     // Arrange
18     var response = await ProgramTest.NewClient.GetAsync("api/catalog-items/0");
19     Assert.AreEqual(HttpStatusCode.NotFound, response.StatusCode);
20 }
21
22 [TestMethod]
23 public void ReturnsOkGivenValidId()
24 {
25 }
```

Request

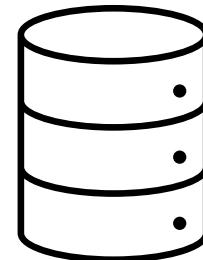


Request

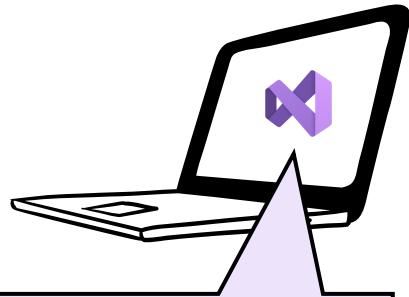
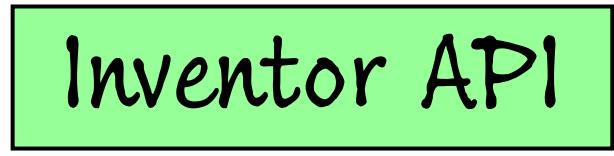


Response

Response



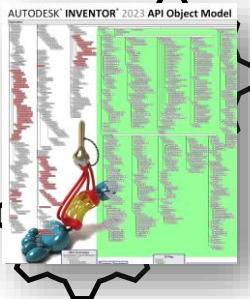
# Understanding what an API is (Inventor Add-in Example)



```
14 [TestMethod]
15 public async Task ReturnsNotFoundGivenInvalidId()
16 {
17     var response = await ProgramTest.NewClient.GetAsync("api/catalog-items/0");
18     Assert.AreEqual(HttpStatusCode.NotFound, response.StatusCode);
19 }
20
21 [TestMethod]
22 public void ReturnsOkGivenValidId()
23 {
24     var response = await ProgramTest.NewClient.GetAsync("api/catalog-items/1");
25 }
```

A code editor window showing a snippet of C# test code using the Microsoft Test Framework and the Microsoft HttpClient class to make requests to an API endpoint.

Request

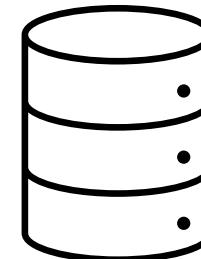


Request

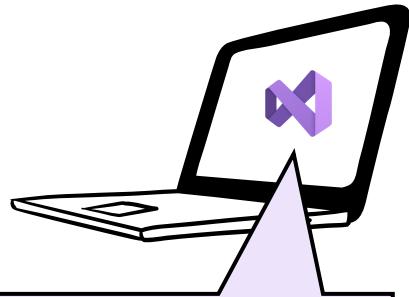
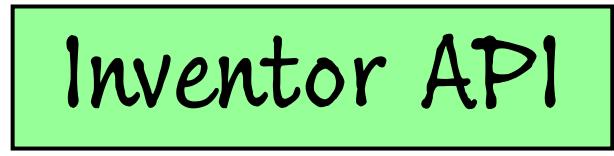


Response

Response



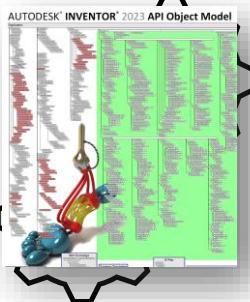
# Understanding what an API is (Inventor Add-in Example)



A dark code editor window showing a snippet of C# code related to the Inventor API.

```
14 [TestMethod]
15 public void TestCatalogItemsChange()
16 {
17     using (var client = new Client())
18     {
19         var response = await client.GetAsync("api/catalog-items/0");
20         Assert.AreEqual(HttpStatusCode.NotFound, response.StatusCode);
21     }
22 }
23
24 }
```

Request



Request



Response

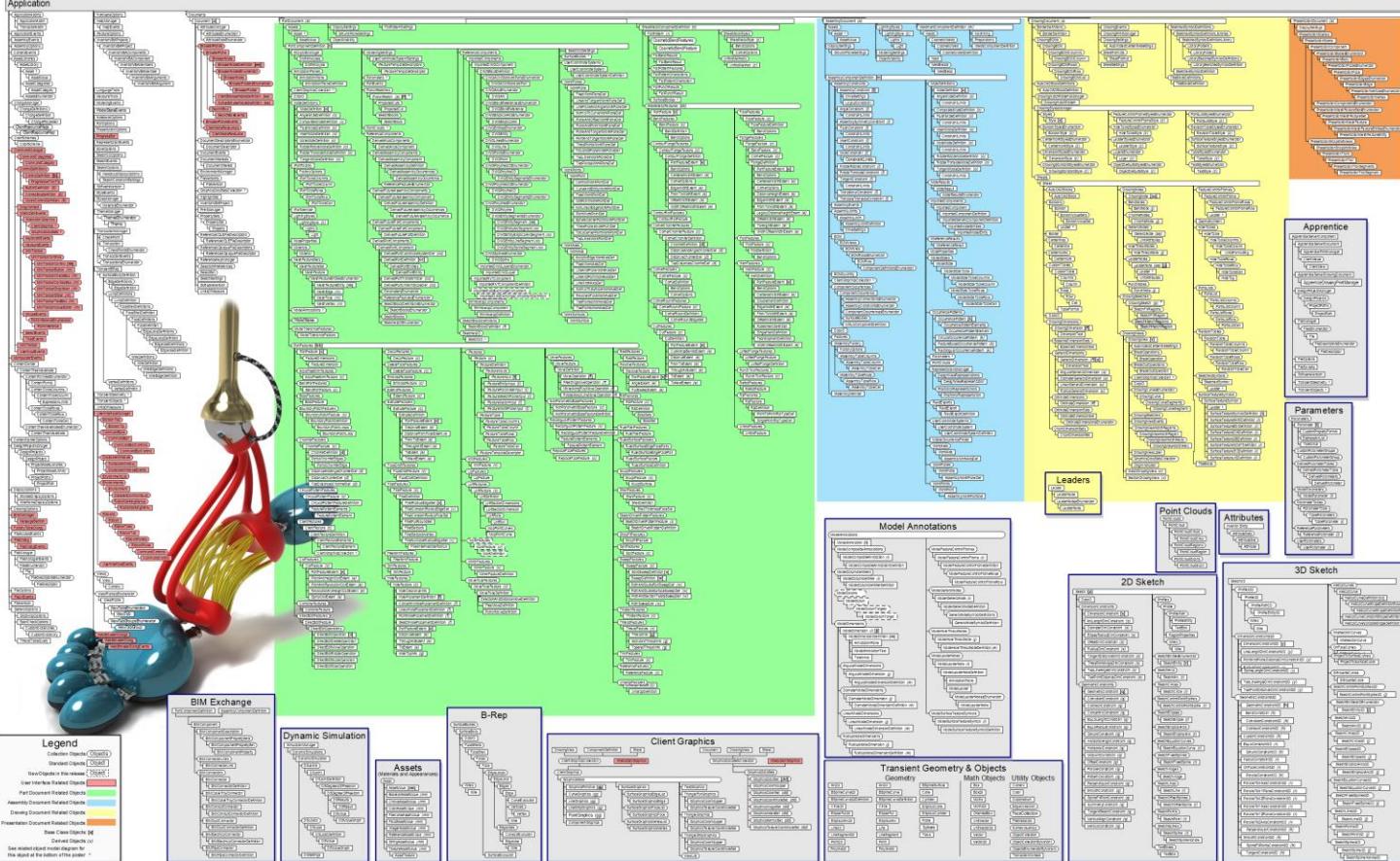


Response



# Understanding the API

## AUTODESK® INVENTOR® 2023 API Object Model



January 10, 2022



# Understanding the API

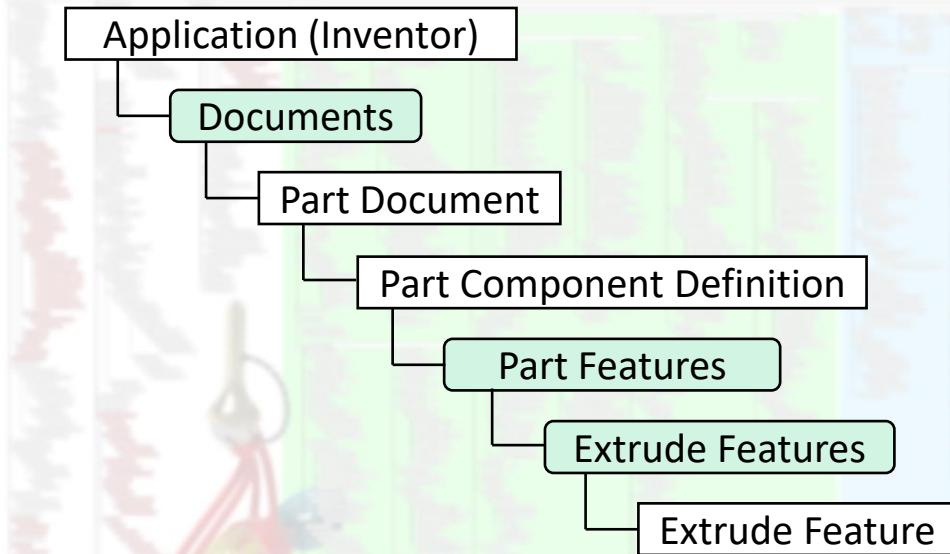
Collections

Object

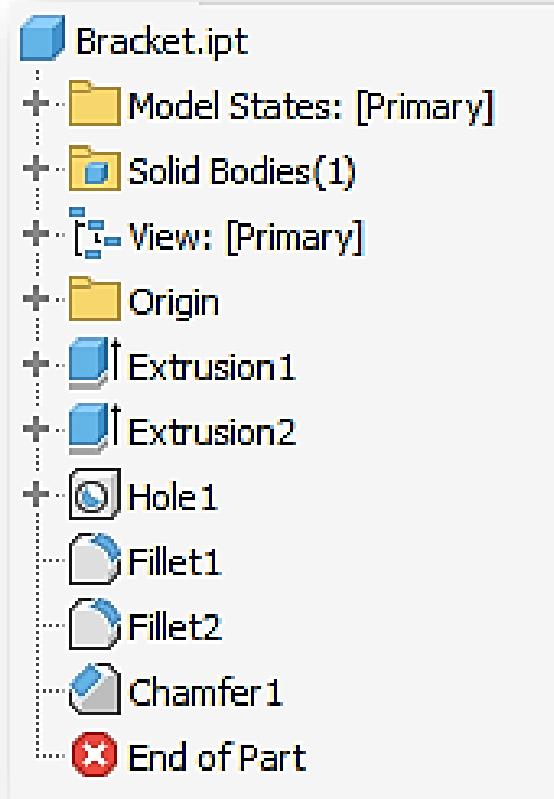
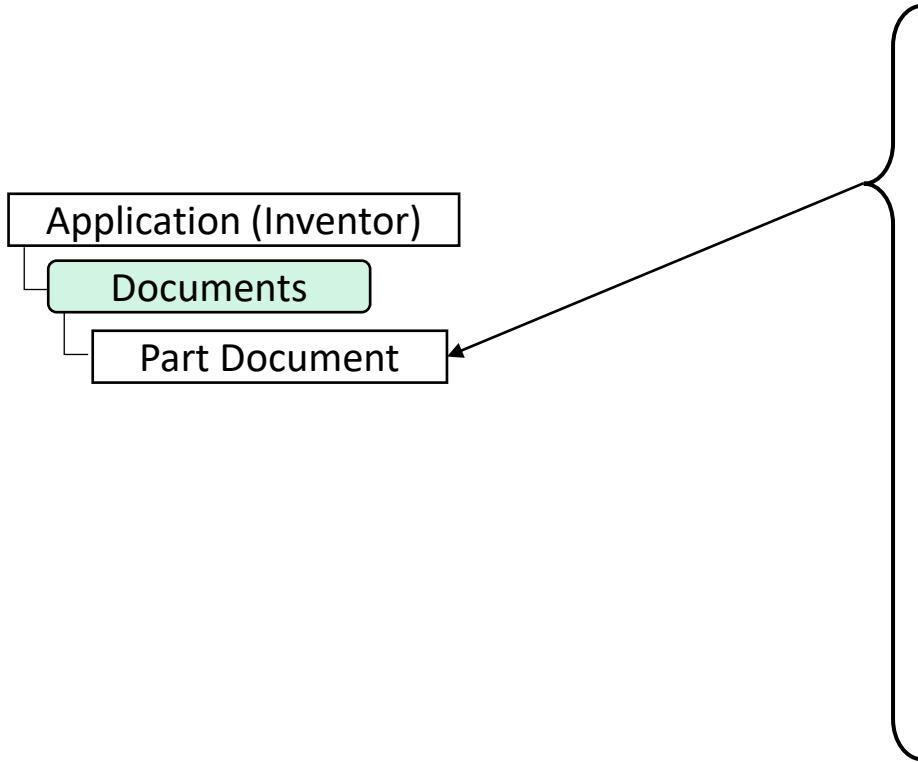
Lunch time...  
Returning at 1:00



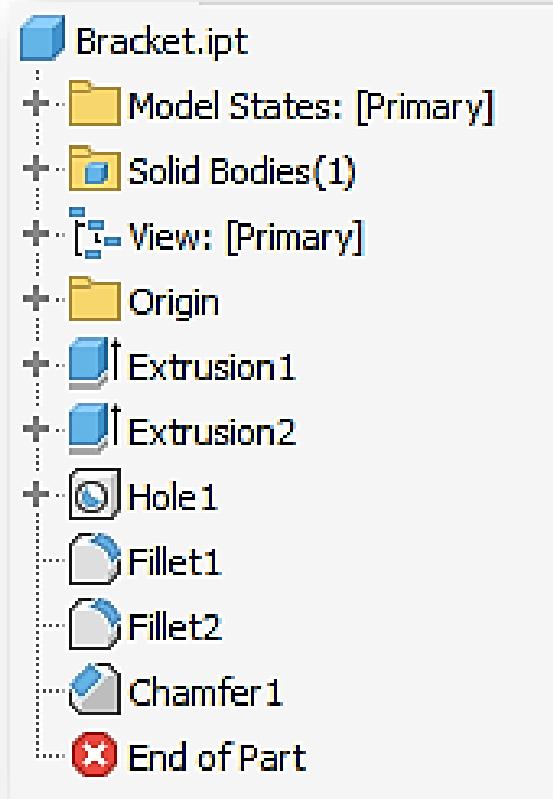
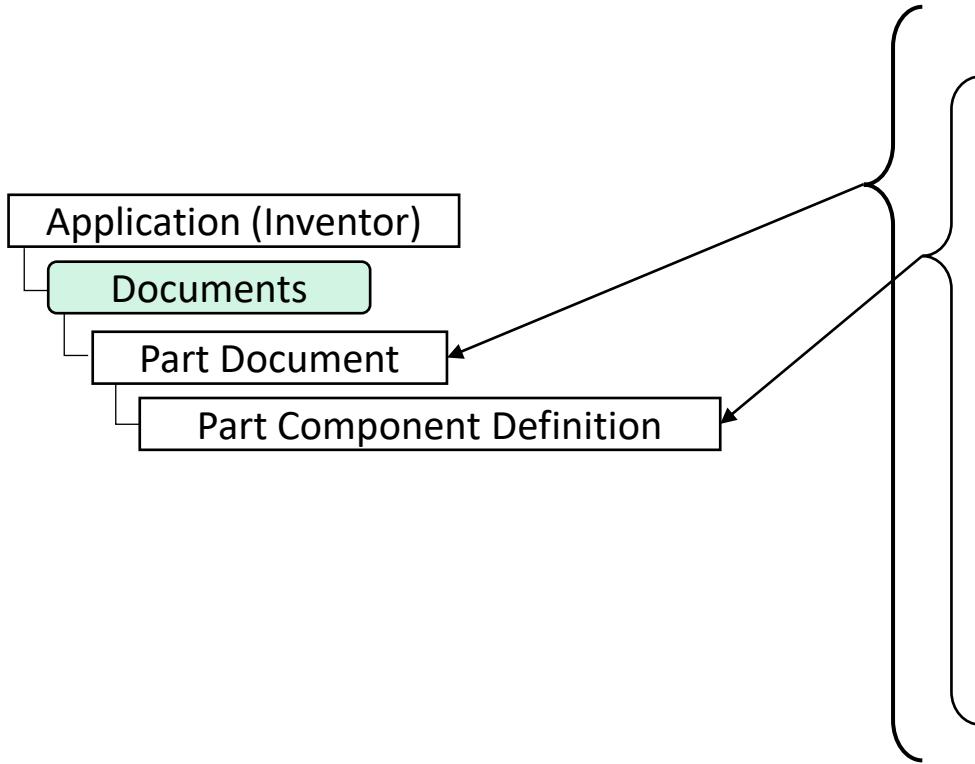
# Understanding the API



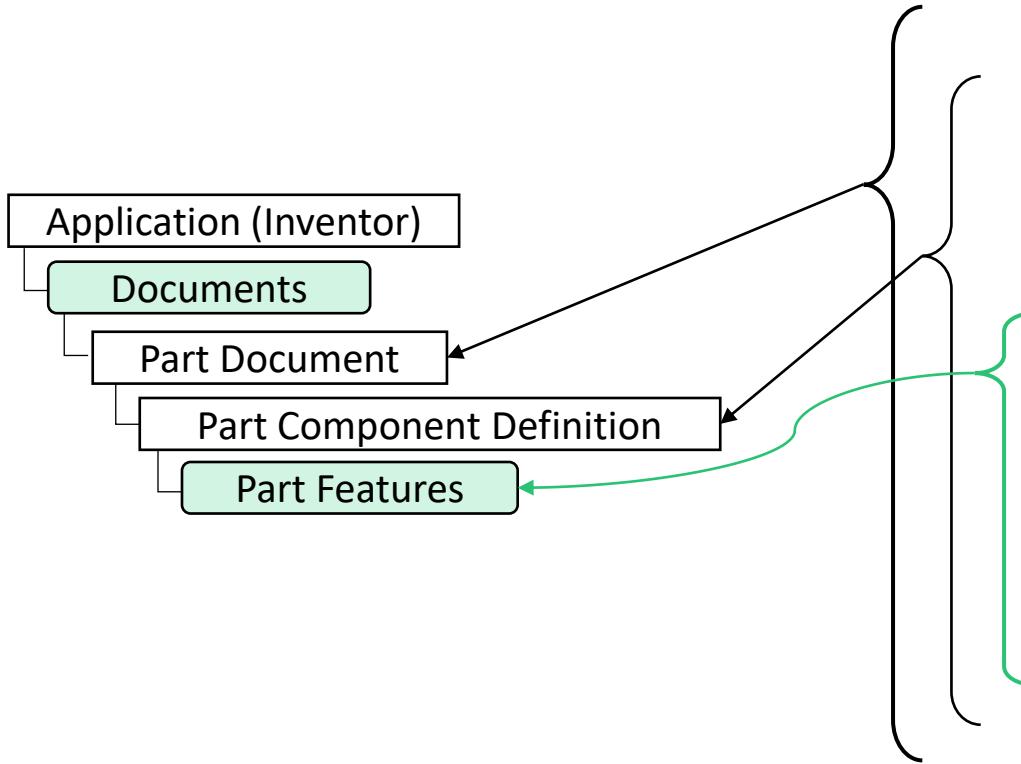
# API Object Model of a part



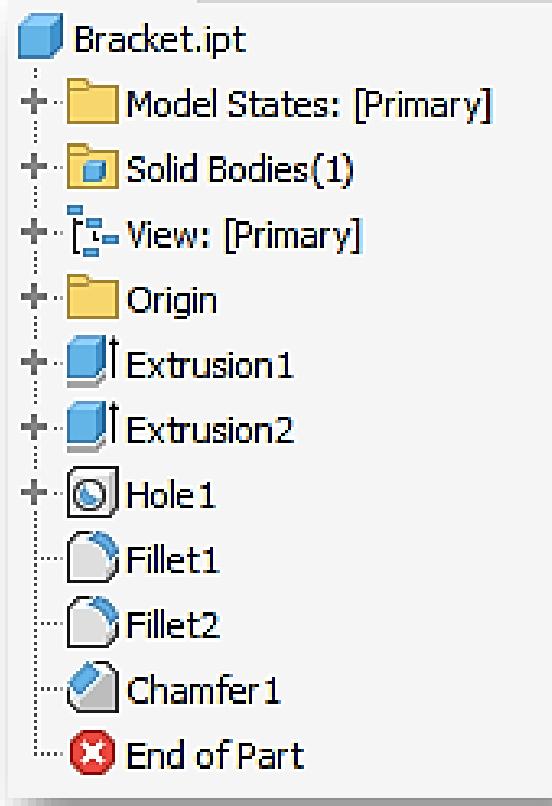
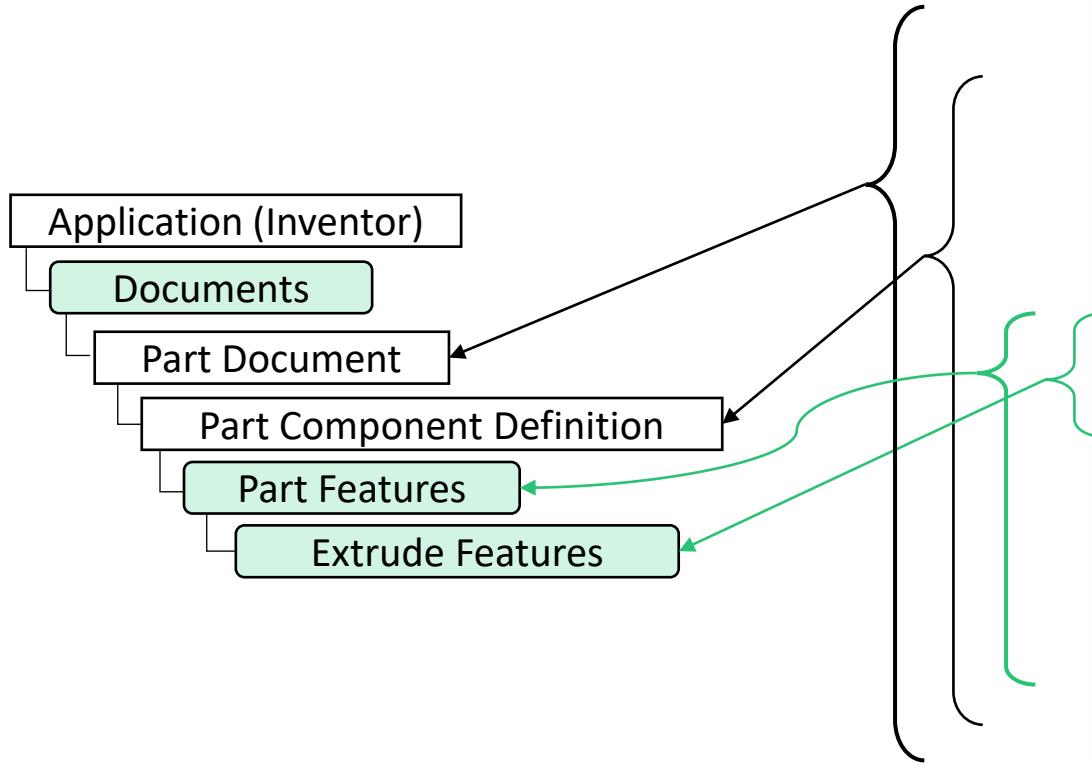
# API Object Model of a part



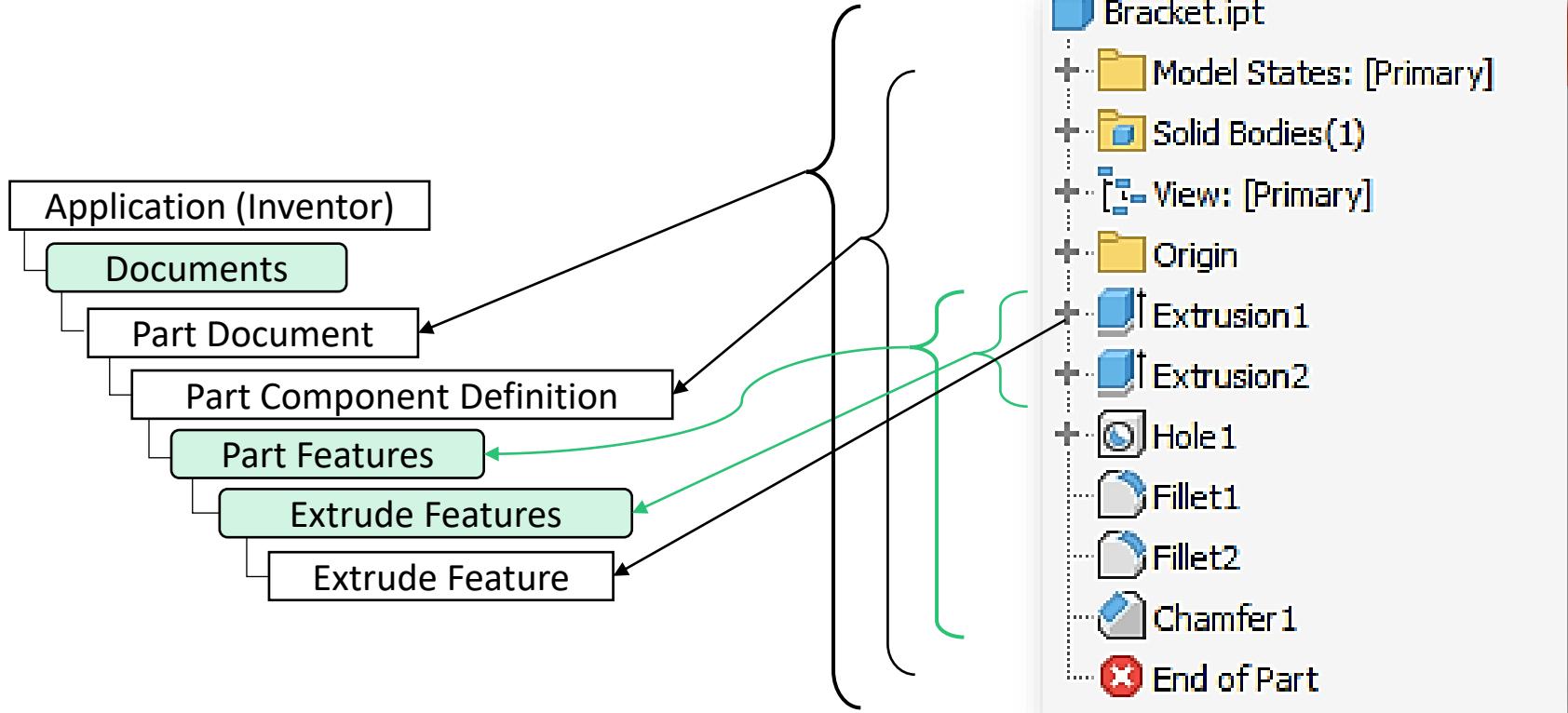
# API Object Model of a part



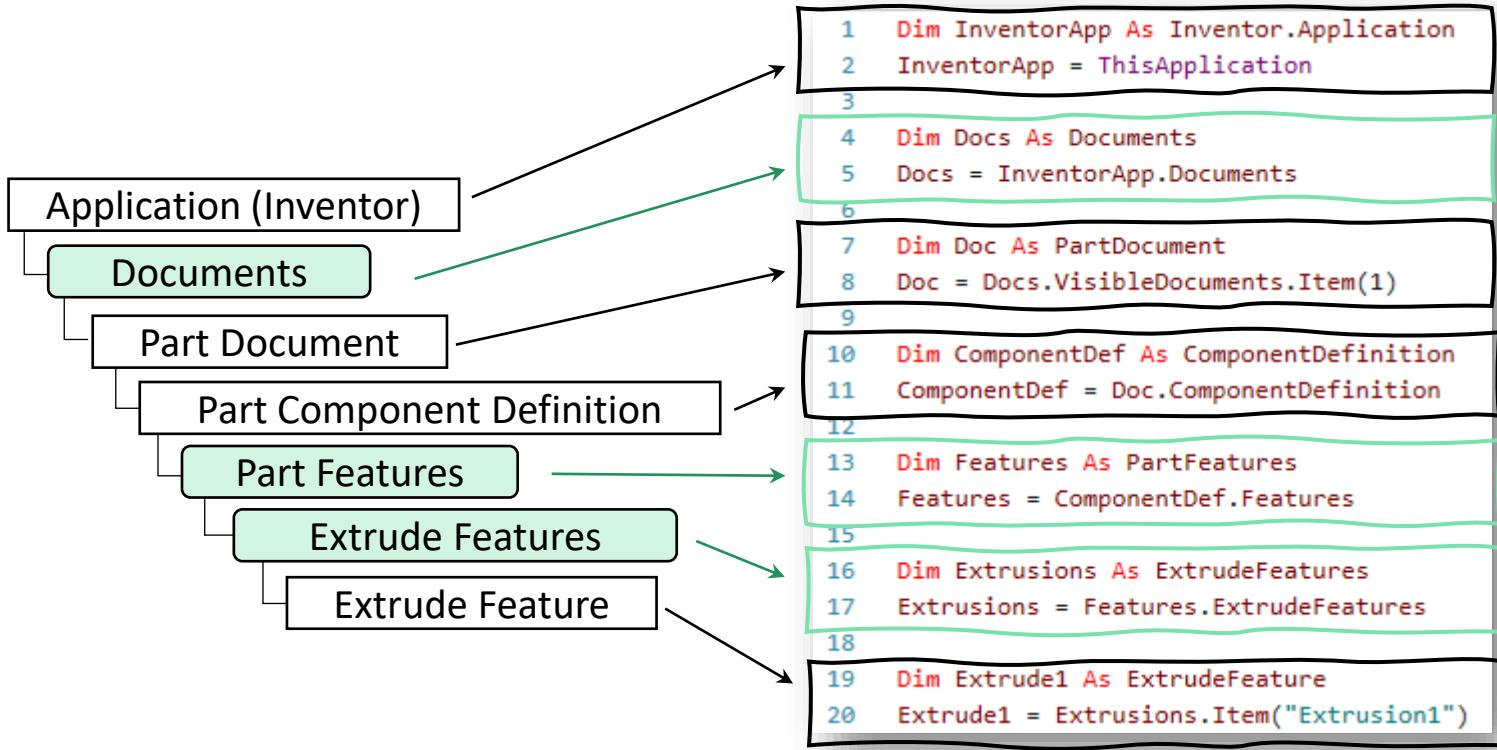
# API Object Model of a part



# API Object Model of a part



# Code Comparison

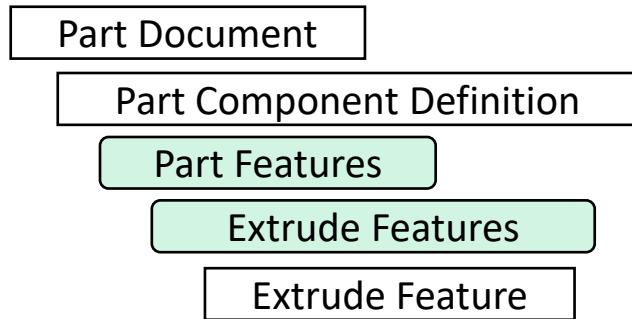


iLogic Example

# Using a single line to get the named Extrusion



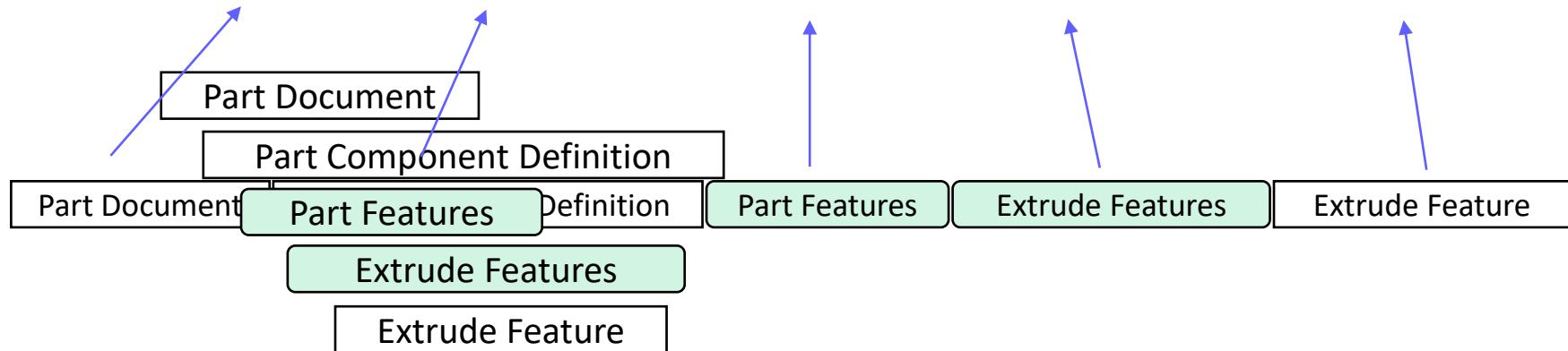
```
Extrude1 = Doc.ComponentDefinition.Features.ExtrudeFeatures.Item("Extrusion1")
```



# Using a single line to get the named Extrusion



```
Extrude1 = Doc.ComponentDefinition.Features.ExtrudeFeatures.Item("Extrusion1")
```





Wait... what  
is this API  
you speak of?

got it!



## SDK = Software Development Kit

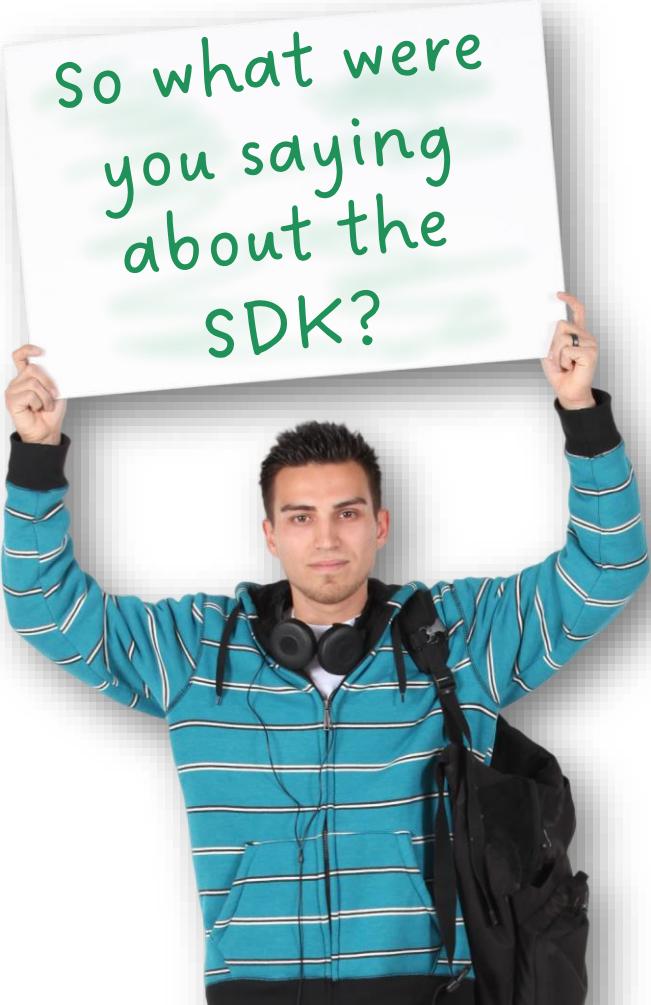
The Inventor SDK includes samples and tools to demonstrate and describe the functionality that is available in the Inventor API\*

\* API = Application Programming Interface

END  
DETOUR

API

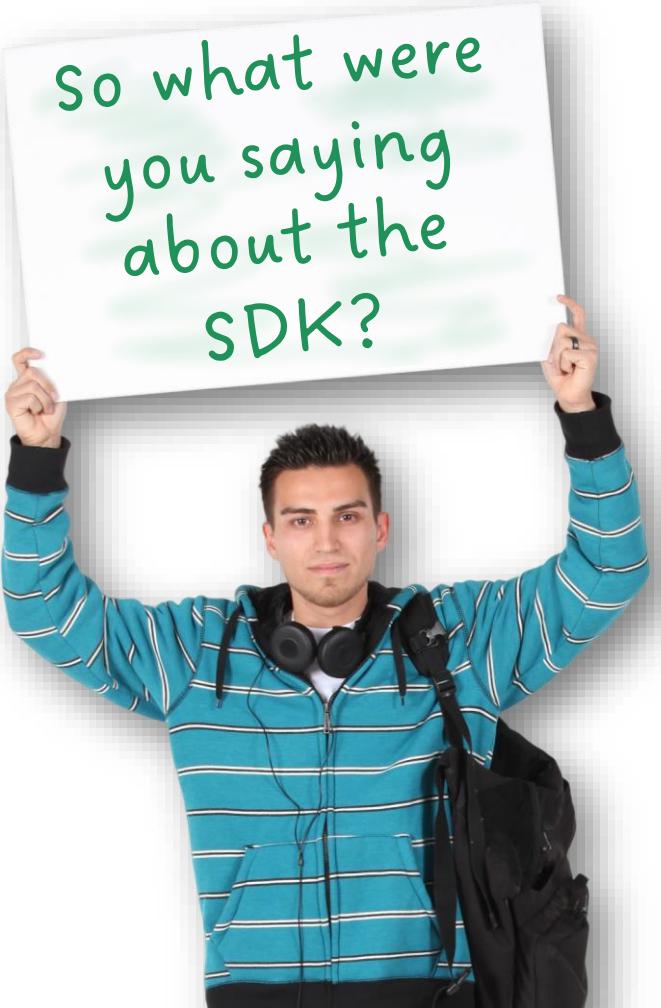
DETOOK



## **SDK = Software Development Kit**

The Inventor SDK includes **samples and tools to demonstrate and describe the functionality that is available in the Inventor API\***

\* API = Application Programming Interface

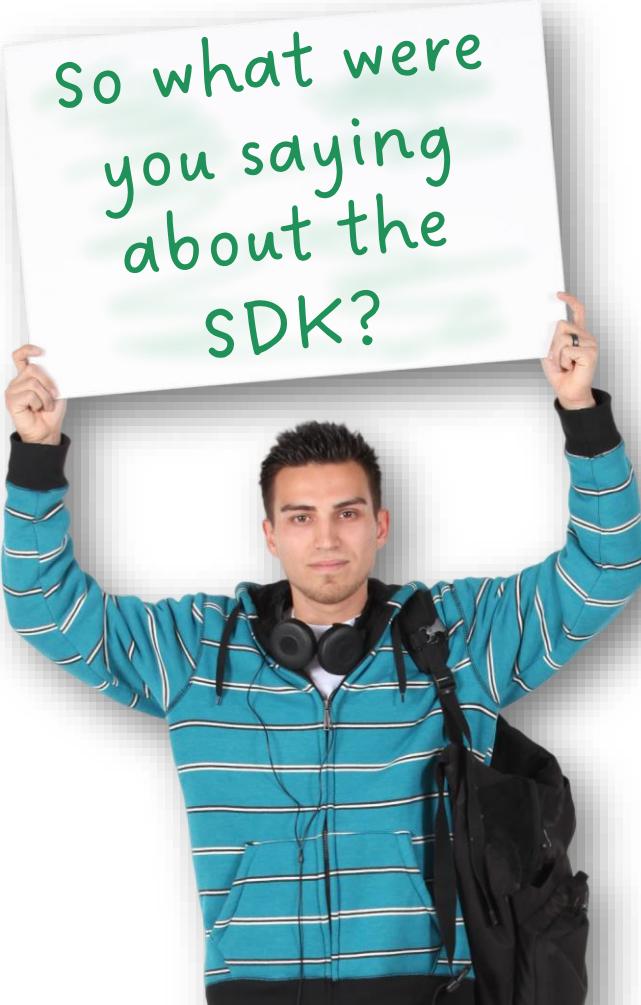


## **SDK = Software Development Kit**

The Inventor SDK includes **samples and tools to demonstrate and describe the functionality that is available in the Inventor API\***

\* API = Application Programming Interface

The SDK installs with Inventor, but we need to “unpack” it



## SDK = Software Development Kit

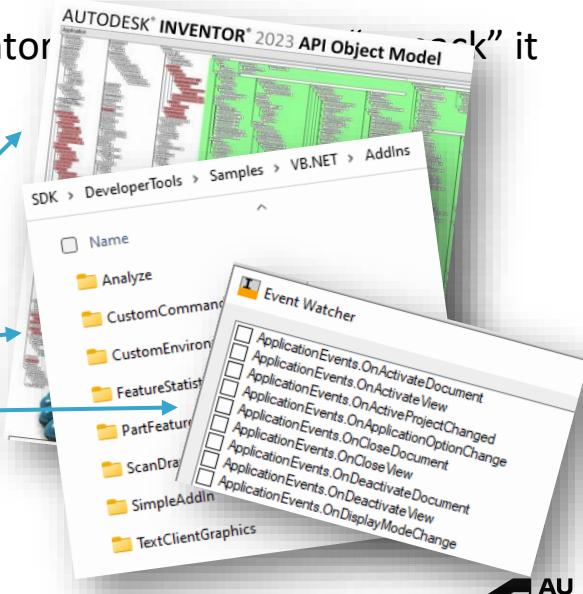
The Inventor SDK includes **samples and tools to demonstrate and describe the functionality that is available in the Inventor API\***

\* API = Application Programming Interface

The SDK installs with Inventor

The SDK includes:

- Developer Tools
  - Documentation
  - Samples
  - Tools
  - Add-in Templates
- User Tools





So what were you saying about the SDK?

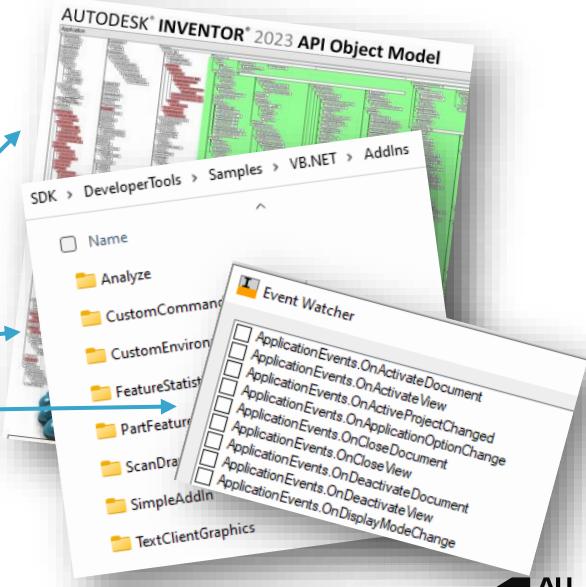
# **SDK = Software Development Kit**

The Inventor SDK includes **samples and tools to demonstrate and describe the functionality that is available in the Inventor API\***

\* API = Application Programming Interface

## The SDK includes:

- Developer Tools
    - Documentation
    - Samples
    - Tools
    - Add-in Template
  - User Tools



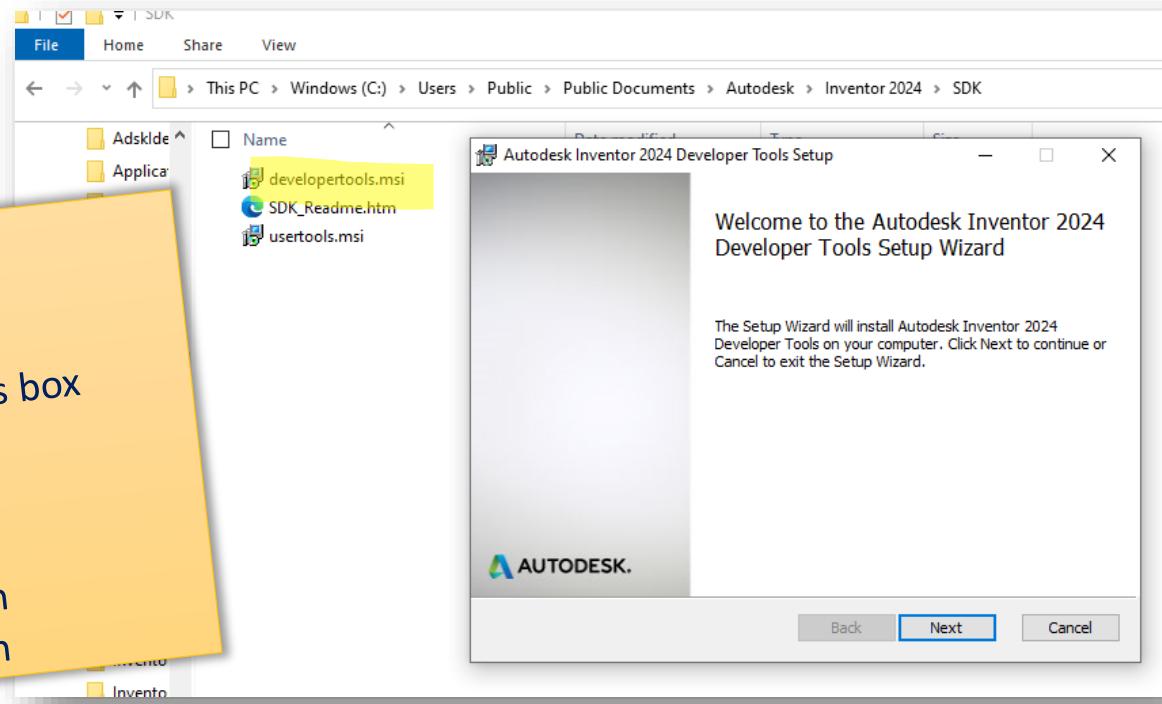
# Unpacking/installing the Inventor SDK files

( Software Developer Kit )



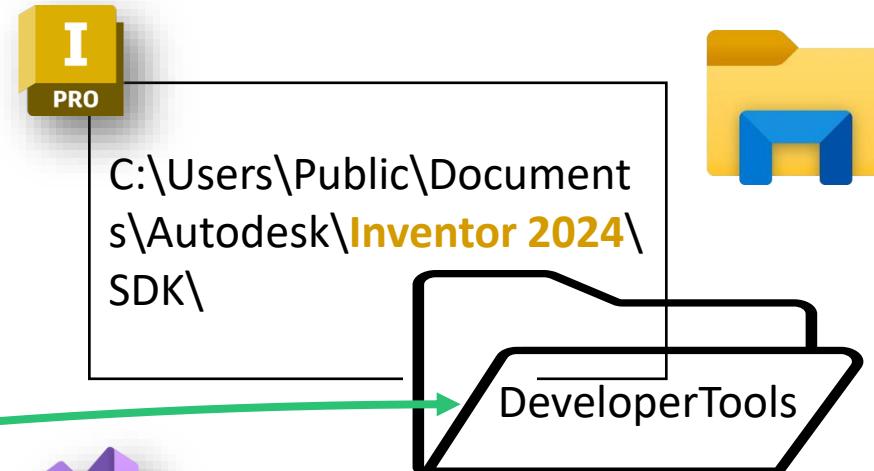
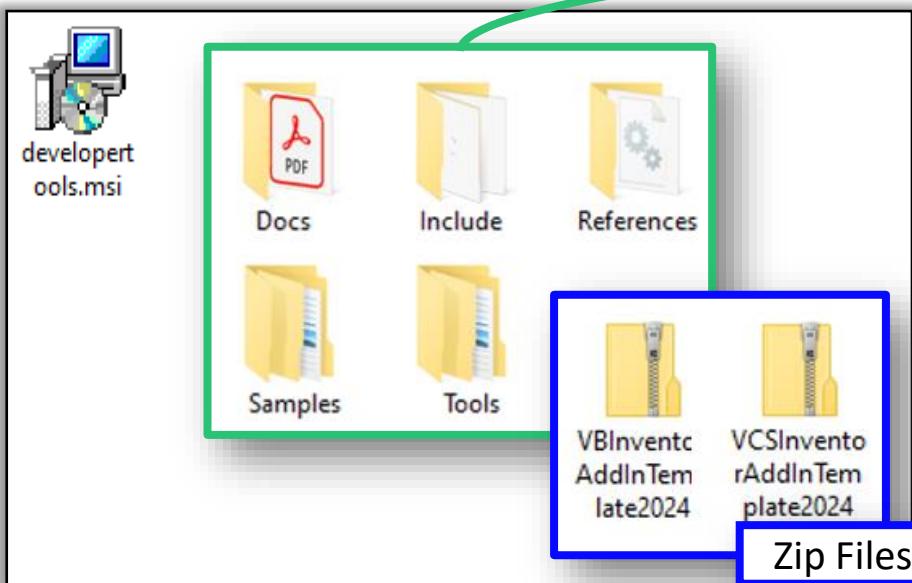
In Windows Explorer go to: C:\Users\Public\Documents\Autodesk\Inventor 2024\SDK

- 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



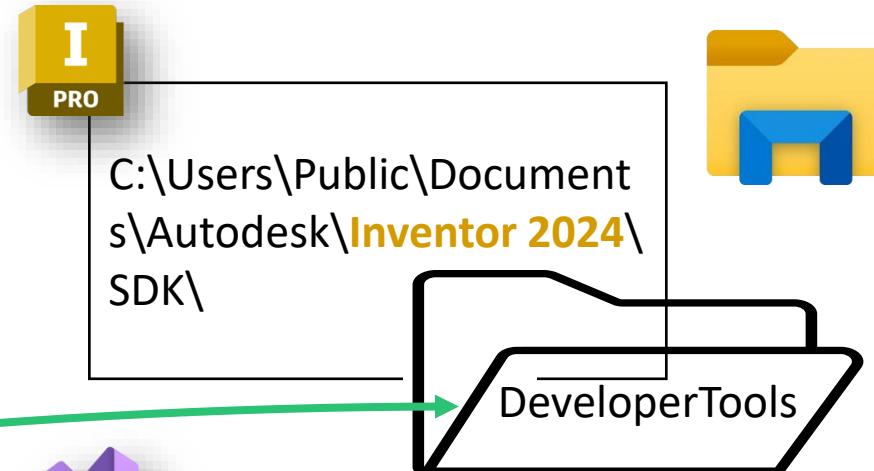
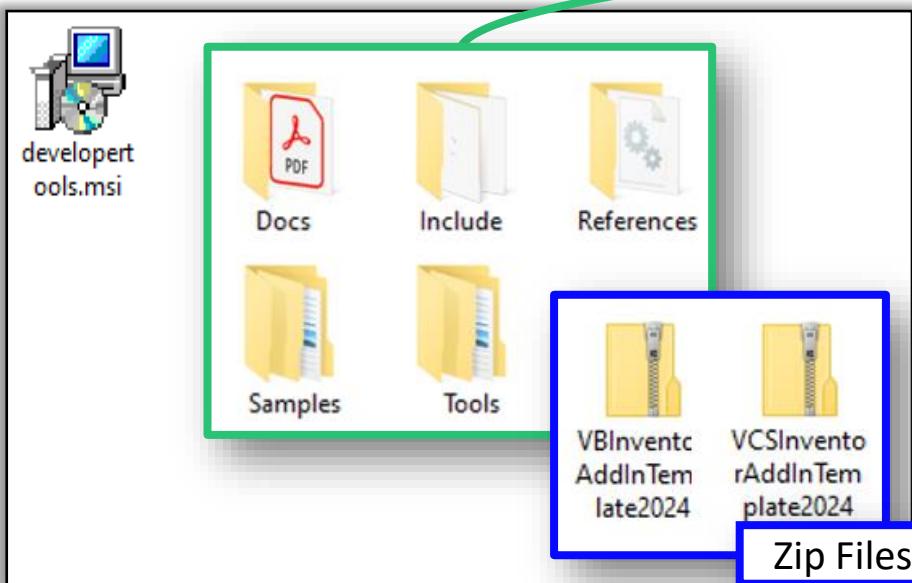
# developertools.msi Contents

Contents of developertools.msi are unpacked into two Locations



# developertools.msi Contents

Contents of developertools.msi are unpacked into two Locations



# Add-in Templates



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



C#



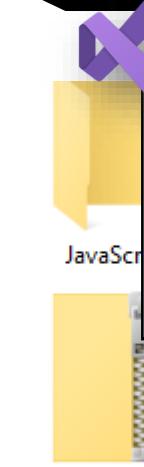
Extensibility



Visual Basic



Visual Web  
Developer



JavaScript

VBlnventorAddIn  
Template2024.zip



VCSInventorAddIn  
Template2024.zip

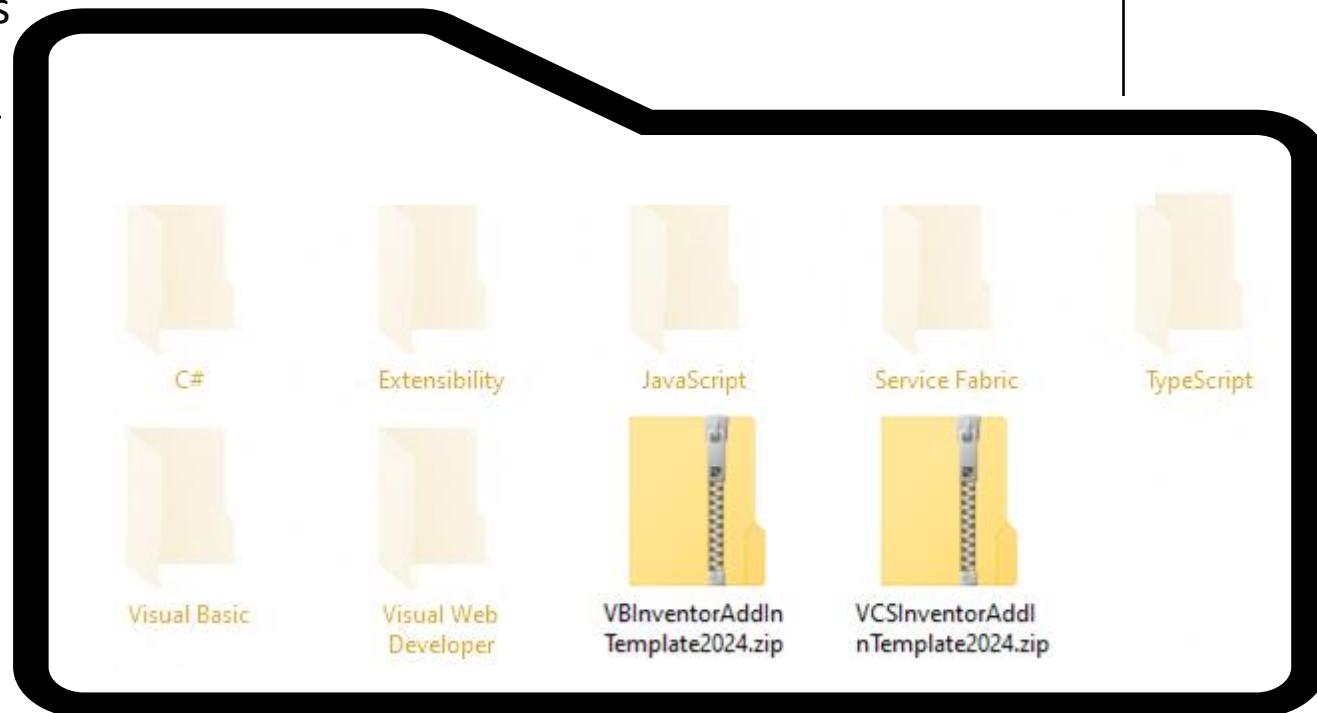
C:\Users\CurtisWaguespack  
Documents\Visual Studio  
2022\Templates\

ProjectTemplates

# Add-in Templates



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



# Steps to get started...

... with Visual Studio and the Inventor SDK



- Install Visual Studio



- Extract the Inventor SDK ( software development kit)



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

# Steps to get started...

... with Visual Studio and the Inventor SDK



- Install Visual Studio



- Extract the Inventor SDK ( software development kit)



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





# Check/set the template location in Visual Studio

Open Visual Studio > click Continue without code

The screenshot shows the Visual Studio 2022 start window. On the left, there's a 'Get started' sidebar with four options: 'Clone a repository', 'Open a project or solution', 'Open a local folder', and 'Create a new project'. Below this is a large 'Continue without code →' button, which is circled in red. At the bottom of the sidebar, there's a link to 'Choose a project template with code scaffolding to get started'. On the right side of the window, there's a 'Recent' section with pinned items and a 'Today' section. At the bottom, there's a feedback survey and a 'Do you like this start window?' question.

Visual Studio 2022

Open recent

Pinned

- D3\_Inventor\_Drawing\_Tools.sln 8/12/2023 10:43 AM C:\...\3D\_Automation\_Services\Drawing\_Tools\_Add-in\Sandbox
- Inventor\_Drawing\_Tools.sln 8/9/2023 3:07 PM C:\...\Inventor\_Drawing\_Tools\Inventor\_Drawing\_Tools\Inventor...
- GTL\_Model\_States\_Tools.sln 8/4/2023 3:11 PM C:\...\GreenPoint\...\Inventor>Addins\...\VSA Macro\GTL\_Model\_S...
- GTL\_Export\_XCD\_Workpoints.sln 1/27/2023 12:59 PM C:\...\GreenPoint\GTL\_Workpoint\_Export\GTL\_Export\_XCD\_Workpoints

Today

- Inventor4dinit.sln 8/12/2023 8:12 AM C:\...\OneDrive - 3D Technologies\Desktop\Throw Away\VS test\In...

This week

Do you like this start window?

Get started

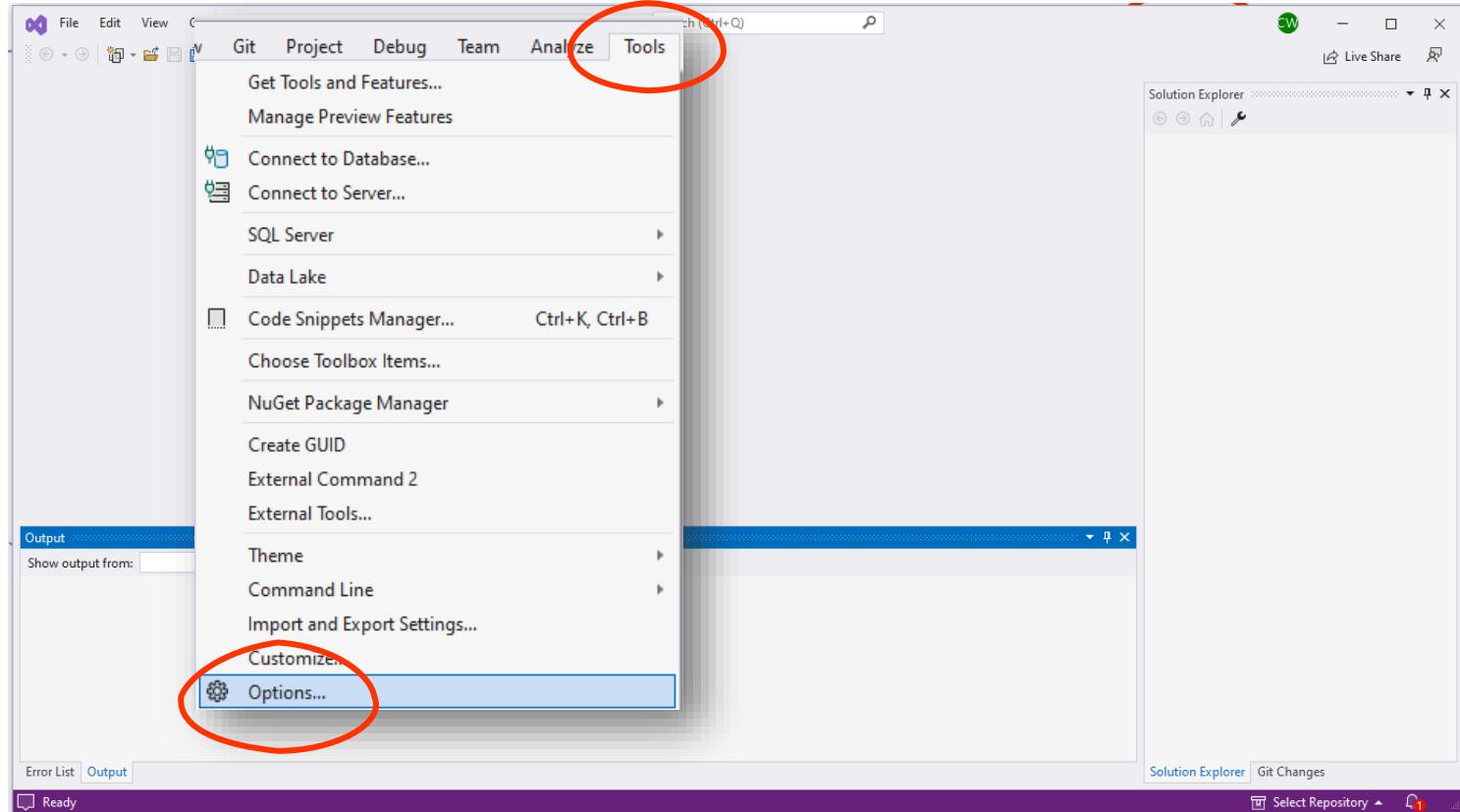
- Clone a repository  
Get code from an online repository like GitHub or Azure DevOps
- Open a project or solution  
Open a local Visual Studio project or .sln file
- Open a local folder  
Navigate and edit code within any folder
- Create a new project  
Choose a project template with code scaffolding to get started

Continue without code →



# Check/set the template location in Visual Studio

Visual Studio > Tools tab > Options





# Check/set the template location in Visual Studio

Visual Studio > Tools tab > Options > Projects and Solutions > Locations

The screenshot shows the 'Options' dialog box in Visual Studio. The left pane lists various settings categories under 'Environment'. Two red arrows point to the 'Locations' category, which is currently selected and highlighted in grey. The right pane displays three template locations:

- Project location:** C:\Users\CurtisWaguespack\source\repos
- User project template location:** C:\Users\CurtisWaguespack\Documents\Visual Studio 2022\Templates\ProjectTemplates (This path is highlighted with a yellow background)
- User item template location:** C:\Users\CurtisWaguespack\Documents\Visual Studio 2022\Templates\ItemTemplates

At the bottom right of the dialog are 'OK' and 'Cancel' buttons.

*You can change this path if you prefer. Or leave it as is.*

*I generally leave it as is.*



# Check/set the template location in Visual Studio

Visual Studio > Tools tab > Options > Projects and Solutions > Locations

The screenshot shows the Visual Studio Options dialog box. In the left pane, under the 'Environment' category, 'Projects and Solutions' is expanded, and 'Locations' is selected. The right pane displays three template locations:

- Project location:** C:\Users\CurtisWaguespack\source\repos
- User project template location:** C:\Users\CurtisWaguespack\Documents\Visual Studio 2022\Templates\ProjectTemplates (This path is highlighted with a yellow box.)
- User item template location:** C:\Users\CurtisWaguespack\Documents\Visual Studio 2022\Templates\ItemTemplates

A yellow sticky note with the following text is overlaid on the dialog:

**TIP:**  
Even if you don't change  
the path, click the OK  
button to create the  
standard folders.

In the bottom right corner of the dialog, there are 'OK' and 'Cancel' buttons. A hand is pointing towards the 'OK' button.

# Check/set the template location in Visual Studio



A woman in a business suit is pointing her right index finger towards the 'OK' button in a modal dialog box. She has a warm smile on her face. The background shows the Visual Studio interface with a 'ProjectTemplates' folder open. A yellow sticky note is overlaid on the bottom left of the dialog box with handwritten text: 'standard location'. The Visual Studio ribbon bar is visible at the top, showing 'File', 'Edit', 'View', 'Project', 'Tools', 'Help'. The 'Project' tab is selected. The 'Solution Explorer' and 'Git Changes' tabs are also visible at the bottom.

# Check/set the template location in Visual Studio

A woman in a black blazer and red top is pointing her right index finger towards a 'Select Repository' dialog box in the background. The dialog box shows the path 'C:\Program Files\Microsoft Visual Studio\2022\Preview\Templates\ProjectTemplates'.

The screenshot shows the 'ProjectTemplates' folder in the Visual Studio Explorer. The left sidebar shows a tree view with 'ProjectTemplates' selected. The main area displays several project templates: VBInventorAddInTemplate2024.zip, VCSIInventorAddInTemplate2024.zip, C#, Extensibility, JavaScript, Service Fabric, TypeScript, Visual Basic, and Visual Web Developer. A small note at the bottom left says 'standard location'.

ProjectTemplates

New |剪切 |复制 |粘贴 |插入 |删除 |排序 |视图 |更多

Templates < Visual Studio 2022 > Templates > ProjectTemplates

Search ProjectTe... |

Templates

ItemTemplates

ProjectTemplates

Downloads

Dropbox

Favorites

Learning

Links

Microsoft

9 items

VBInventorAddInTemplate2024.zip

VCSIInventorAddInTemplate2024.zip

C#

Extensibility

JavaScript

Service Fabric

TypeScript

Visual Basic

Visual Web Developer

standard location

OK Cancel

Error List Output

Solution Explorer Git Changes

Select Repository Ready



# Add-in Templates

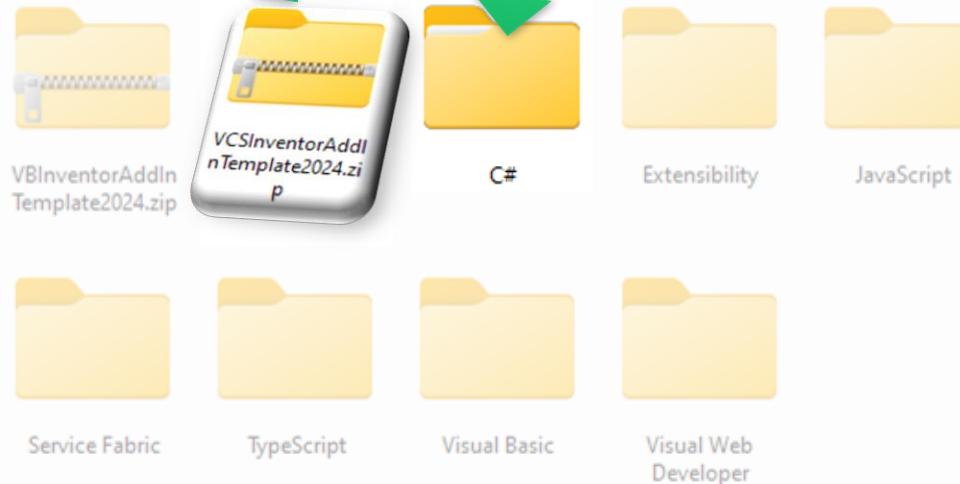


C:\Users\CurtisWaguespack\Documents\Visual Studio 2022\Templates\  
ProjectTemplates

Move the zip file to the C#  
folder.

VCSInventorAddinTemplate

You don't need to unzip it



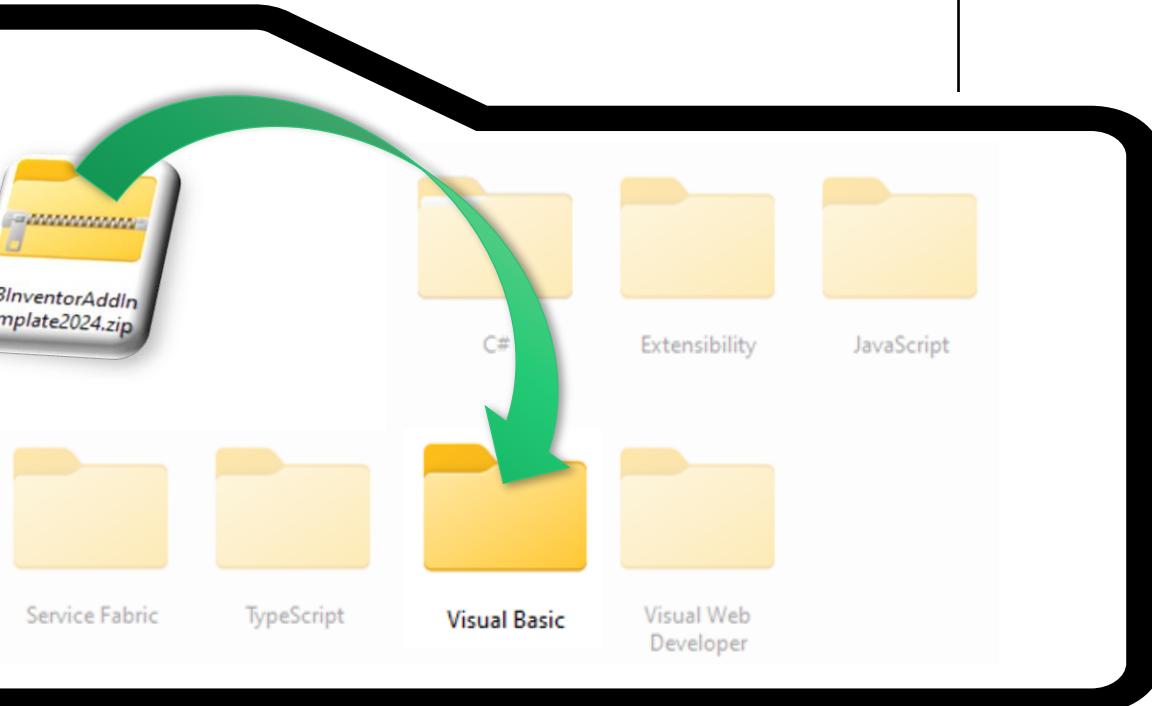


# Add-in Templates



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

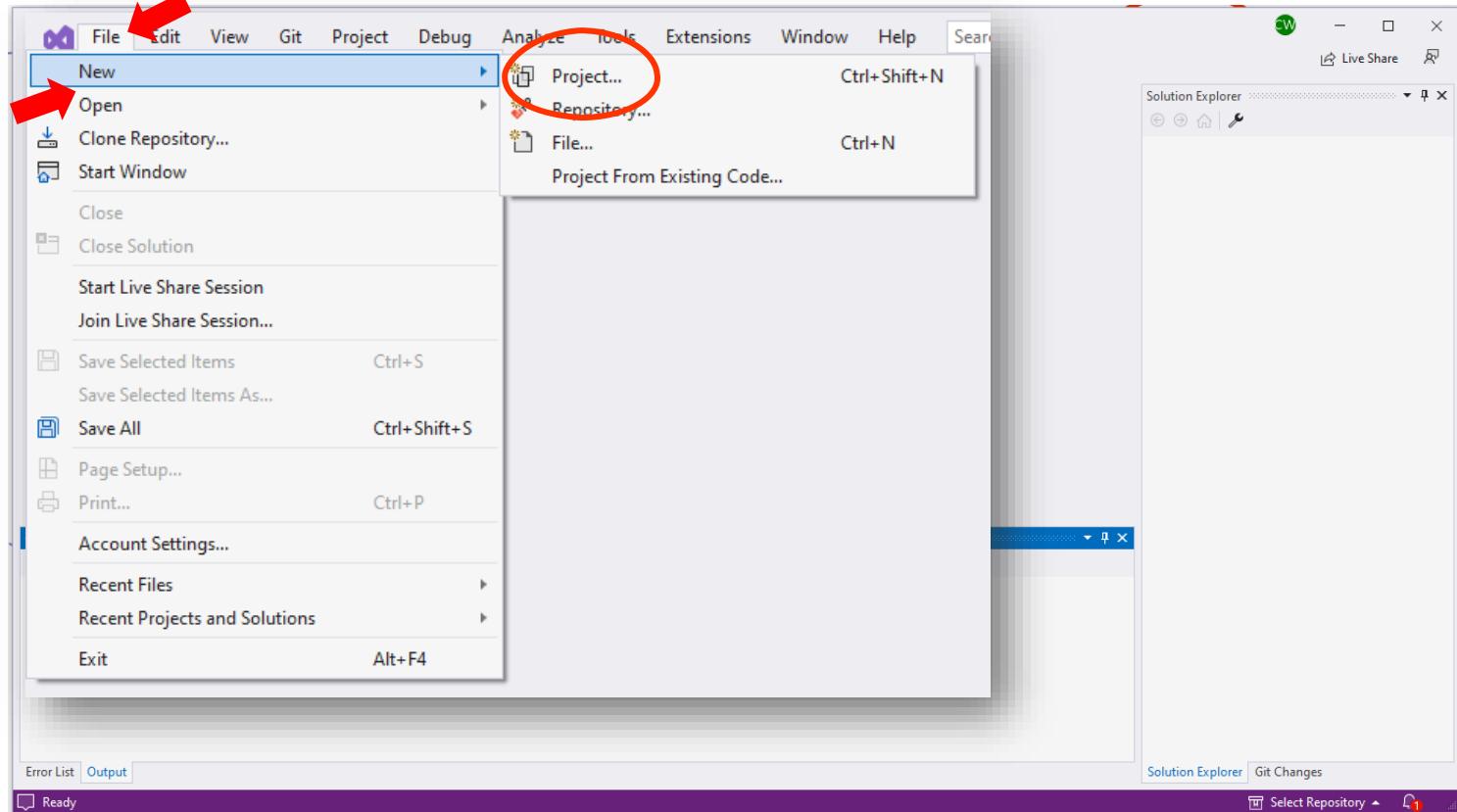
Move the zip file to the  
Visual Basic folder.  
VBInventorAddinTemplate  
You don't need to unzip it





# Check/set the template location in Visual Studio

Visual Studio > File tab > New > Project





# Ensure Add-In Templates show up in Visual Studio

Click Create a New Project

Visual Studio 2022

Open recent

- Pinned
  - D3\_Inventor\_Drawing\_Tools.sln 8/12/2023 10:43 AM C:\...\3D\_Automation\_Services\Drawing\_Tools Add-in\Sandbox
  - Inventor\_Drawing\_Tools.sln 8/9/2023 3:07 PM C:\...\Inventor\_Drawing\_Tools\Inventor\_Drawing\_Tools\Inventor...
  - GTL Model States Tools.sln 8/4/2023 3:11 PM C:\...\GreenPoint\...\Inventor>Addins\...\VSA Macro\GTL\_Model\_S...
  - GTL Export ICD Workpoints.sln 1/27/2023 12:59 PM C:\...\GreenPoint\GTL Workpoint Export\GTL Export ICD\Workpoints
- Today
  - InventorAddin.sln 8/12/2023 8:12 AM C:\...\OneDrive - 93 Technologies\Desktop\Throw Away\VS test\In...
- This week

Get started

- Clone a repository
- Open a project or solution
- Open a local folder
- Create a new project

Do you like this start window?

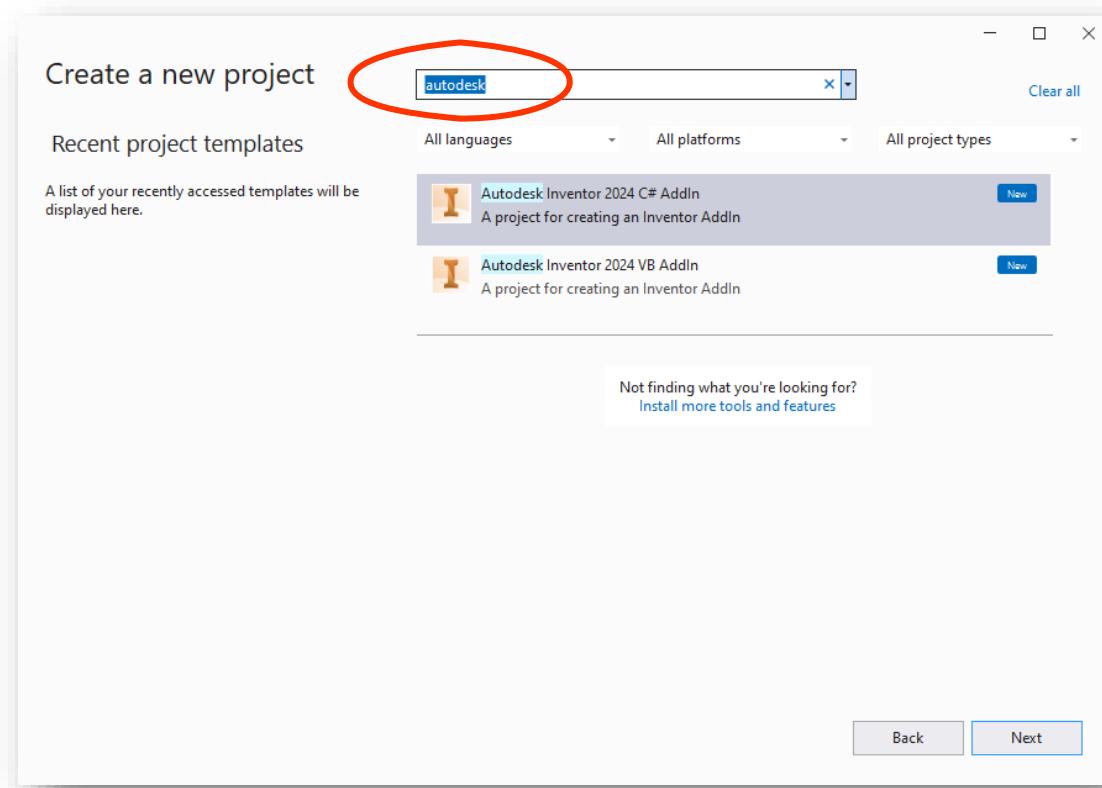
[Continue without code →](#)



# Ensure Add-In Templates show up in Visual Studio



- Type Autodesk or Inventor in the search bar
- You should see the templates show up, as shown in this image.



# Now you're ready to create an add-in!



-  • Install Visual Studio
-  • Extract the Inventor SDK ( software development kit)
-  • Ensure the Inventor Add-In Templates show up in Visual Studio

# Now you're ready to create an add-in!



Except  
you're not!

-  Install Visual Studio
-  Extract the Inventor SDK ( software development kit)
-  Ensure the Inventor Add-In Templates show up in Visual Studio

# Quick Review



Review

# Steps to get started...

... with Visual Studio and the Inventor Software Development Kit



**1**



Install  
Visual  
Studio

**2**



Install  
the  
Inventor  
SDK

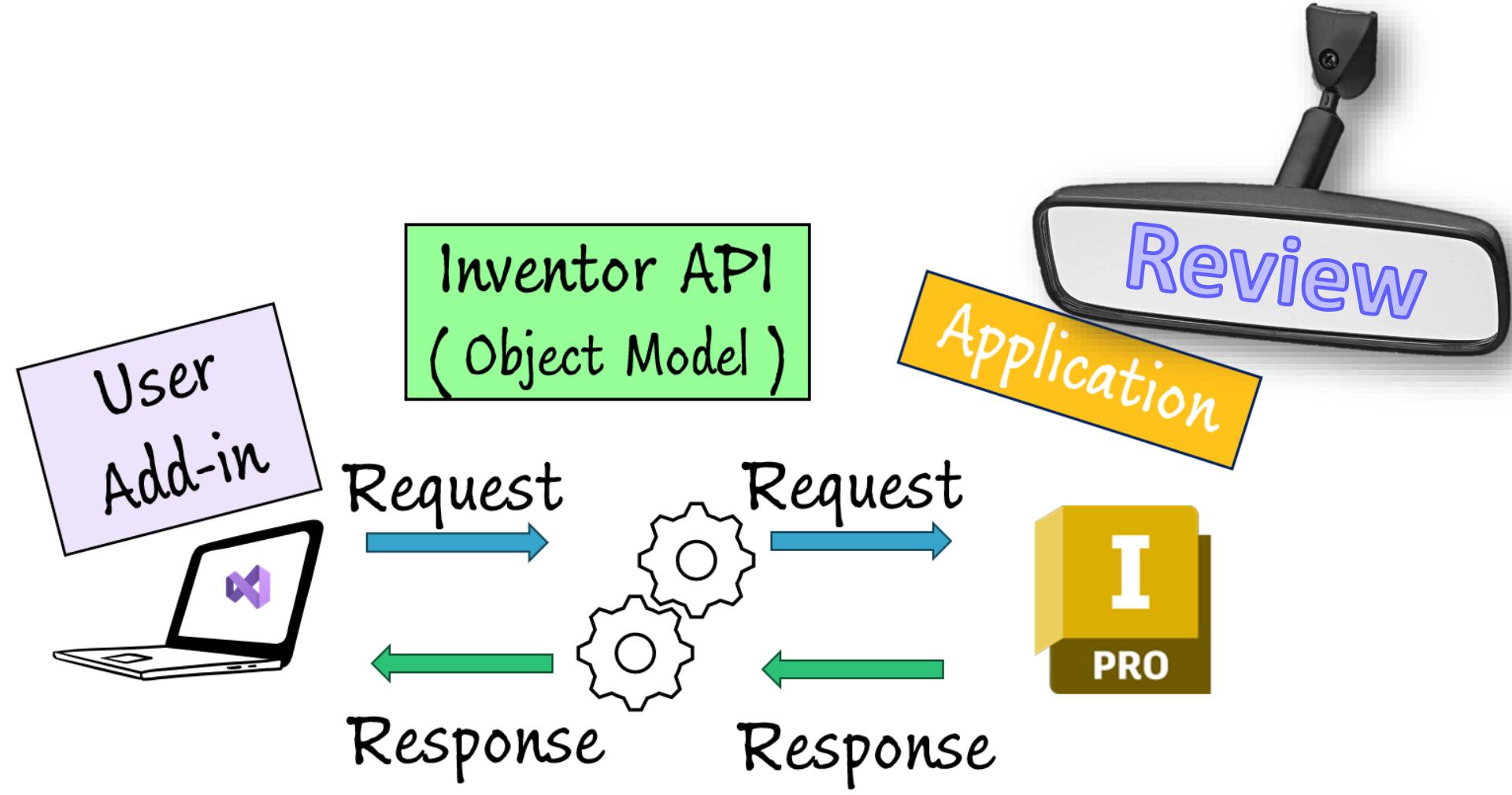
**3**



Ensure the  
Inventor  
Add-in  
Templates  
are Loaded

**Review**





# Review

Application (Inventor)

Documents

Part Document

Part Component Definition

Part Features

Extrude Features

Extrude Feature

Bracket.ipt

+ Model States: [Primary]

+ Solid Bodies (1)

+ View: [Prim]

+ Origin

+ Extrusion1

+ Extrusion2

+ Hole1

+ Fillet1

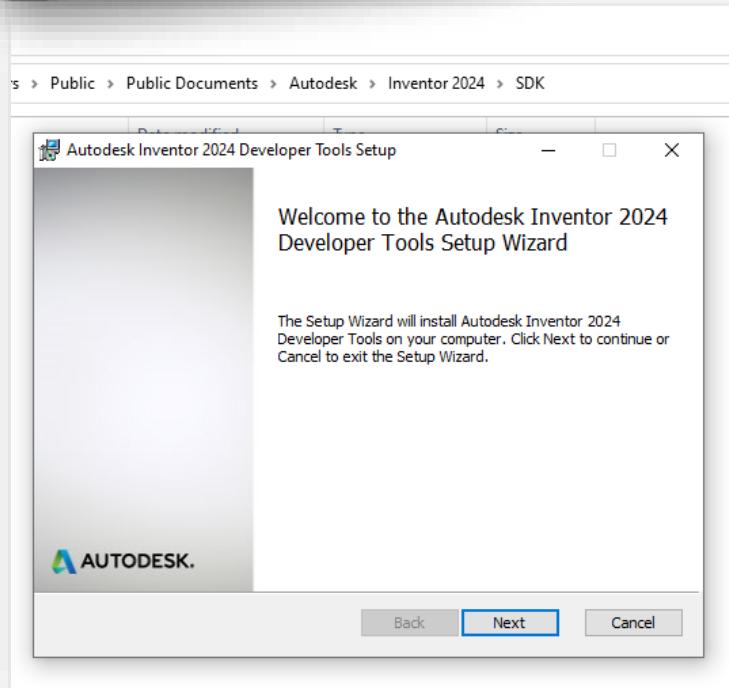
+ Fillet2

+ Chamfer1

- End of Part

```
1 Dim InventorApp As InventorApp
2 InventorApp = ThisApplication
3
4 Dim Docs As Documents
5 Docs = InventorApp.Documents
6
7 Dim Doc As PartDocument
8 Doc = Docs.VisibleDocuments(1)
9
10 Dim ComponentDef As ComponentDefinition
11 ComponentDef = Doc.ComponentDefinition
12
13 Dim Features As PartFeatures
14 Features = ComponentDef.Features
15
16 Dim Extrusions As Extrusions
17 Extrusions = Features.Extrusions
18
19 Dim Extrude1 As Extrusion
20 Extrude1 = Extrusions(1)
```

# Review



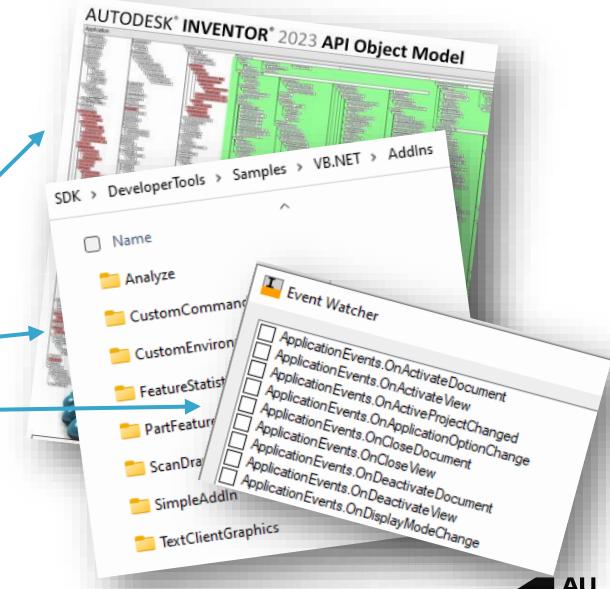
## SDK = Software Development Kit

The Inventor SDK includes **samples and tools to demonstrate and describe the functionality that is available in the Inventor API\***

\* API = Application Programming Interface

The SDK includes:

- Developer Tools
  - Documentation
  - Samples
  - Tools
  - Add-in Templates
- User Tools



# An Improved Add-in Template

Downloading the files



# Getting the custom templates

## Download the Class Materials

The screenshot shows a session page from Autodesk University. At the top, there's a navigation bar with the Autodesk logo, 'AUTODESK UNIVERSITY', and links for 'Agenda & Experience', 'Sessions', 'Expo', 'P...', and 'FAQ'. Below the navigation, there's a search bar with placeholder text 'Search for keywords in videos, presentation slides and handouts:' and a magnifying glass icon. To the right of the search bar is a large yellow sticky note with handwritten text: 'The Autodesk University page might or might not look like this at some later date when you are watching this video.' and 'But should look close'. On the left side of the main content area, there's a sidebar with icons for 'Overview' (document), 'Video' (camera), 'Presentation' (document with a grid), and 'Downloads' (down arrow). A red arrow points to the 'Downloads' button. The main content area features a session title 'MFG601910 | Bridging the Gap between iLogic Automation and Inventor Add-Ins' with a small icon of two people. Below the title, it says 'Tuesday, Nov 14 | 10:30 AM - 12:00 PM PST'. Underneath the title is a 'Description' section with a paragraph of text about iLogic automation. To the right of the description is a 'Speakers' section featuring a photo of a man with a beard, Curtis Waguespack, and the text 'Curtis Waguespack Automation Solution Consultant'.

AUTODESK

AUTODESK UNIVERSITY Agenda & Experience Sessions Expo P... FAQ

Search for keywords in videos, presentation slides and handouts:

MFG601910 | Bridging the Gap between iLogic Automation and Inventor Add-Ins

Tuesday, Nov 14 | 10:30 AM - 12:00 PM PST

**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. Join this session and other Inventor automation-focused users like yourself, and you'll discover a path forward that allows you to capitalize on the familiarity of your existing automation, while learning the landscape of the Inventor add-in.

**Speakers**

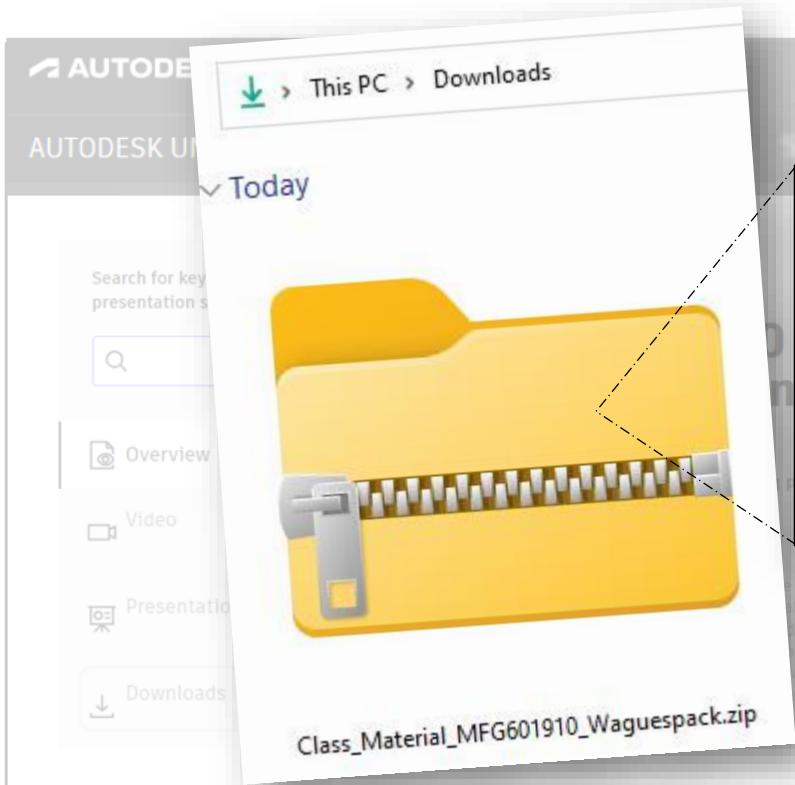
Curtis Waguespack  
Automation Solution Consultant

Downloads

AU 2023

# Finding the add-in files

Find the Templates within the class materials



The screenshot shows the Autodesk UI interface. On the left, there's a sidebar with options like Overview, Video, Presentations, and Downloads. The main area shows a yellow folder icon with a zipper, representing a zip file. The path 'This PC > Downloads' is visible at the top. A callout bubble points to this icon with the text: 'Exact file count and names might vary by the time this recording is released.'

Below the folder icon, the file name 'Class\_Material\_MFG601910\_Waguespack.zip' is displayed.

On the right, a file explorer window is open, showing the contents of the downloaded zip file. The path 'This PC > Downloads > Class\_Material\_MFG601910\_Waguespack.zip' is shown. Inside the folder, there are two sub-folders:

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

A note at the bottom of the slide states: 'Exact file count and names might vary by the time this recording is released.'



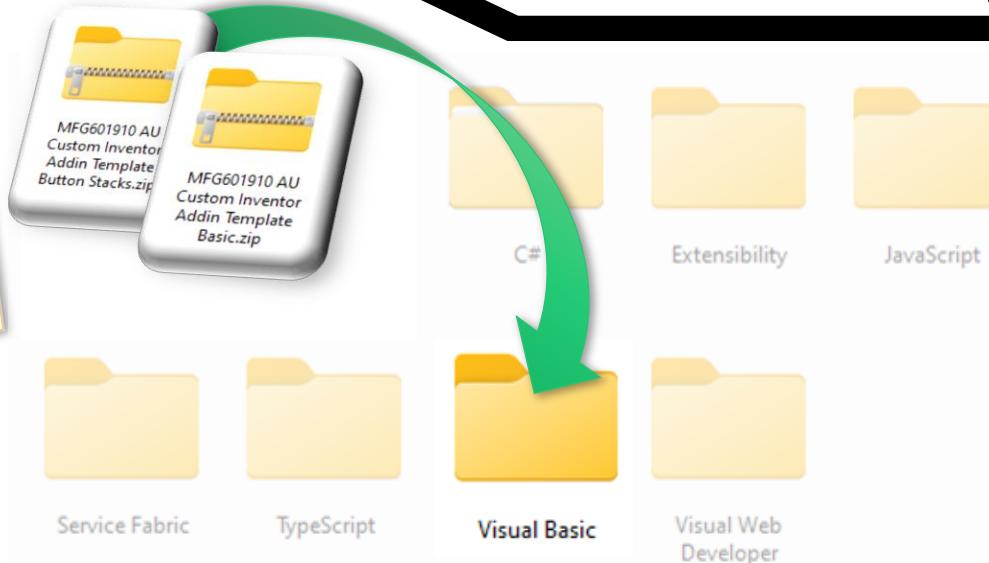
# Add-in Templates



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

Move the template zip files  
to the Visual Basic folder.

You don't need to unzip  
them.



# Creating an Add-in from an improved template

Getting Started





# Creating an Add-in

Click Create a New Project

Visual Studio 2022

Open recent

- Pinned
  - D3\_Inventor\_Drawing\_Tools.sln 8/12/2023 10:43 AM C:\...\3D Automation Services\Drawing Tools Add-in\Sandbox
  - Inventor\_Drawing\_Tools.sln 8/9/2023 3:07 PM C:\...\Inventor\_Drawing\_Tools\Inventor\_Drawing\_Tools\Inventor...
  - GPI\_Model\_Status\_Tools.sln 8/4/2023 3:11 PM C:\...\GreenPoint\...\Inventor Addins and VBA Macro\GPI\_Model\_S...
  - GPI\_Export\_XCD\_Workpoints.sln 1/27/2023 12:59 PM C:\...\GreenPoint\GPI\_Workpoint Export\GPI\_Export\_XCD\_Workpoints
- Today
  - InventorAddin.sln 8/12/2023 8:12 AM C:\...\OneDrive - 93 Technologies\Desktop\Throw Away\VS test\In...
- This week

Get started

- Clone a repository
- Open a project or solution
- Open a local folder
- Create a new project

Do you like this start window?

[Continue without code →](#)

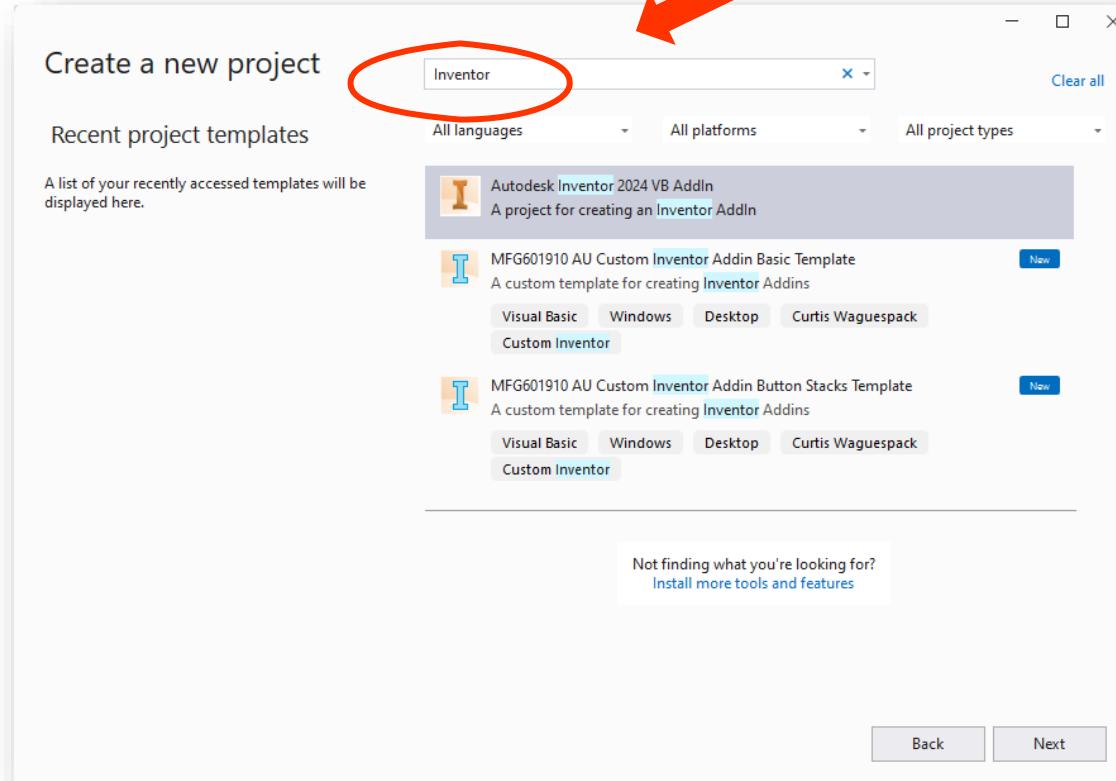




# Creating an Add-in

## 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

Create a new project

Recent project templates

A list of your recently accessed templates will be displayed here.

Search for templates (Alt+S)

All languages All platforms

Curtis Waguespack

MFG601910 AU Custom Inventor Addin Button Stacks Template  
A custom template for creating Inventor Addins  
Visual Basic Windows Desktop Curtis Waguespack  
Custom Inventor

MFG601910 AU Custom Inventor Addin Basic Template  
A custom template for creating Inventor Addins  
Visual Basic Windows Desktop Curtis Waguespack  
Custom Inventor

Not finding what you're looking for?  
[Install more tools and features](#)

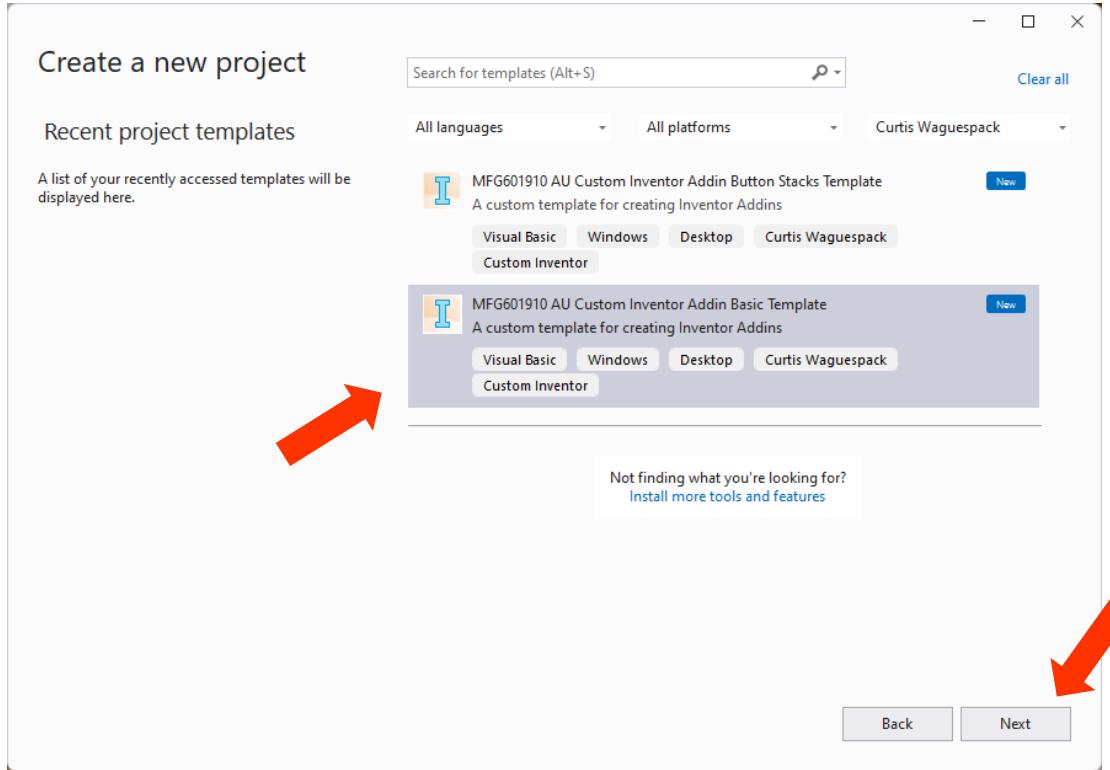
Back Next



# 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

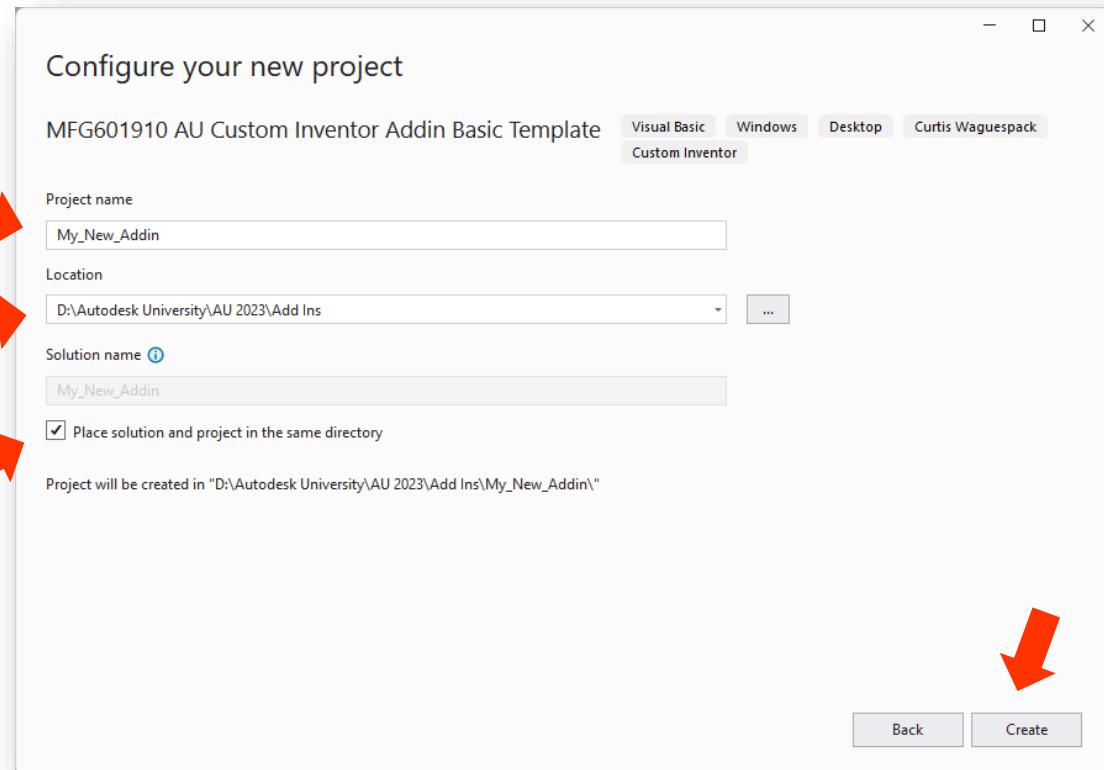




# Creating an Add-in

## 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

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

The screenshot shows the Microsoft Visual Studio interface. The ribbon bar at the top has a yellow box highlighting the "View" tab. Below the ribbon is the "Solution Explorer" window, which displays a solution named "My\_New\_Addin" containing several files and folders. A red arrow points from the woman's hand towards the Solution Explorer window.

File	Edit	View	Git	Project	Build	Debug	Test	Analyze	Tools	Extensions	Window	Help
Solution Explorer	Ctrl+Alt+L		Git Changes	Ctrl+0, Ctrl+G								
Git Repository	Ctrl+0, Ctrl+R		Team Explorer	Ctrl+V, Ctrl+M								
Server Explorer	Ctrl+Alt+S		Data Lake Analytics Explorer									
SQL Server Object Explorer			Test Explorer	Ctrl+T								

Live Share

Add to Source Control Select Repository

# Creating an Add-in



The screenshot shows the Microsoft Visual Studio interface. In the top navigation bar, the 'File' tab is selected. The 'Solution Explorer' window is open, displaying a solution named 'My\_New\_Addin' containing several files and folders. A context menu is open over the solution node, with the 'Open Folder in File Explorer' option highlighted by a red box.

Build Solution  
Rebuild Solution  
Clean Solution  
Analyze and Code Cleanup  
Configuration Manager...  
Manage NuGet Packages for Solution...  
Restore NuGet Packages  
New Solution Explorer View  
Add  
Configure Startup Projects...  
Create Git Repository...  
Paste  
Rename  
Copy Full Path  
**Open Folder in File Explorer** (highlighted)  
Open in terminal  
Save As Solution Filter  
Hide Unloaded Projects  
Properties

Output

Show output from:

Error List Output

- 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 screenshot shows the Microsoft Visual Studio interface with the following details:

- Solution Explorer:** Displays the project structure for "My\_New\_Addin".
- Items in Solution:**
  - Folder: .vs (File folder)
  - Folder: bin (File folder)
  - Folder: Command Tools (File folder)
  - Folder: My Project (File folder)
  - Folder: obj (File folder)
  - Folder: Resources (File folder)
  - Folder: Utility Tools (File folder)
  - File: AssemblyInfo.vb (Visual Basic Source File)
  - File: Autodesk.My\_New\_Addin.Inventor.ad... (ADDIN File)
  - File: My\_New\_Addin.manifest (MANIFEST File)
  - File: My\_New\_Addin.sln (Visual Studio Solution)
  - File: My\_New\_Addin.vbproj (Visual Basic Project File)
  - File: My\_New\_Addin.vbproj.user (Per-User Project Options File)
  - File: Readme.txt (Text Document)
  - File: StandardAddInServer.vb (Visual Basic Source File)
- Error List:** Shows 0 errors.
- Output:** Shows the message "Ready".

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

# Creating an Add-in



The screenshot shows the Microsoft Visual Studio interface. The top menu bar includes File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help, and a Search bar. The toolbar below has icons for Undo, Redo, Cut, Copy, Paste, Find, Replace, and others. The status bar at the bottom shows "Build succeeded" and "Ready".

The main area displays the Solution Explorer, which lists one project named "My\_New\_Add-In". A context menu is open over the project node, with the "Build Solution" option highlighted by a red box and a cursor. Other options in the menu include Rebuild Solution, Clean Solution, Analyze and Code Cleanup, Configuration Manager..., Manage NuGet Packages for Solution..., Restore NuGet Packages, New Solution Explorer View, Add, Configure Startup Projects..., Create Git Repository..., Paste, Rename, Copy Full Path, Open Folder in File Explorer, Open in Terminal, Save As Solution Filter, Hide Unloaded Projects, and Properties.

The Output window below shows the build logs:

```
Show output from: Build
1> INFO: Could not find files for the given pattern(s).
1> The system cannot find the path specified.
1> 'mt.exe' is not recognized as an internal or external command,
1> operable program or batch file.
1> D:\Autodesk University\AU 2023\Add Ins\My_New_Add-In\bin\Debug\My_New_
1> 1 File(s) copied
1> D:\Autodesk University\AU 2023\Add Ins\My_New_Add-In\Autodesk.My_New_Add-In.Inventor.addin
1> 1 File(s) copied
===== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped =====
===== Build started at 4:25 PM and took 00.680 seconds =====
```

The Error List tab is selected in the Output window.

- The solution is shown in the Solution Explorer



# Creating an Add-in



The screenshot shows the Microsoft Visual Studio interface. A yellow sticky note on the left side of the menu bar says: "Tip: you can turn the Output window on/off using the View tab". The menu bar is open, and the "Output" option is highlighted with a blue selection bar. The main window shows the Solution Explorer with a project named "My\_New\_Addin". The Output window at the bottom displays the build logs:

```
Show output from: Build
1> INFO: Could not find files for the given pattern(s).
1> The system cannot find the path specified.
1> 'mt.exe' is not recognized as an internal or external command,
1> operable program or batch file.
1> D:\Autodesk University\AU 2023\Add Ins\My_New_AddIn\bin\Debug\My_New_AddIn.dll
1> 1 File(s) copied
1> D:\Autodesk University\AU 2023\Add Ins\My_New_AddIn\Autodesk.My_New_AddIn.Inventor.addin
1> 1 File(s) copied
===== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped =====
===== Total 1 minute 0 seconds and took 00.723 seconds =====
|
```

A yellow sticky note on the right side of the screen says: "Note the output logging indicates the solution build status." The "Output" tab is selected in the Output window.

- The Build results are shown in the Output window

# Overall Process



# Visual Studio



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



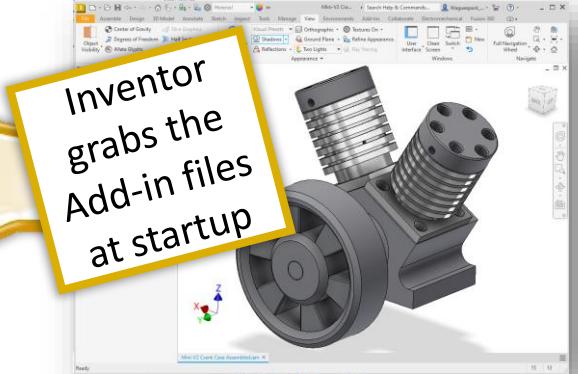
# Autodesk Application Plugins Folder



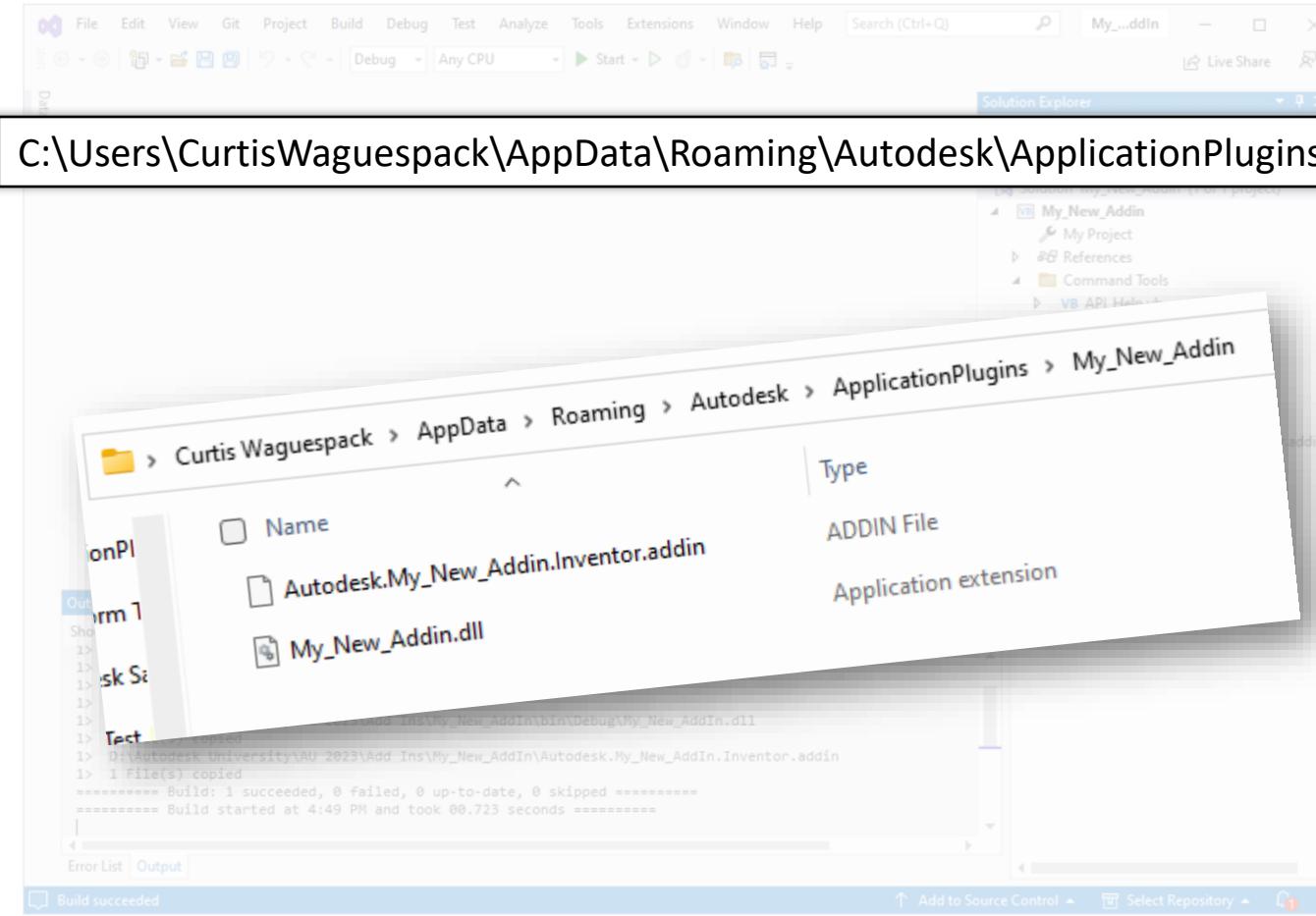
# Review



# Autodesk Inventor



# Creating an Add-in



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

# Creating an Add-in from an improved template

Getting Started





# Creating an Add-in

Click Create a New Project

Visual Studio 2022

Open recent

- Pinned
  - D3\_Inventor\_Drawing\_Tools.sln 8/12/2023 10:43 AM C:\...\3D Automation Services\Drawing Tools Add-in\Sandbox
  - Inventor\_Drawing\_Tools.sln 8/9/2023 3:07 PM C:\...\Inventor\_Drawing\_Tools\Inventor\_Drawing\_Tools\Inventor...
  - GPI\_Model\_Status\_Tools.sln 8/4/2023 3:11 PM C:\...\GreenPoint\...\Inventor Addins and VBA Macro\GPI\_Model\_S...
  - GPI\_Export\_XCD\_Workpoints.sln 1/27/2023 12:59 PM C:\...\GreenPoint\GPI\_Workpoint Export\GPI\_Export\_XCD\_Workpoints
- Today
  - InventorAddin.sln 8/12/2023 8:12 AM C:\...\OneDrive - 93 Technologies\Desktop\Throw Away\VS test\In...
- This week

Get started

- Clone a repository
- Open a project or solution
- Open a local folder
- Create a new project

Do you like this start window?

[Continue without code →](#)

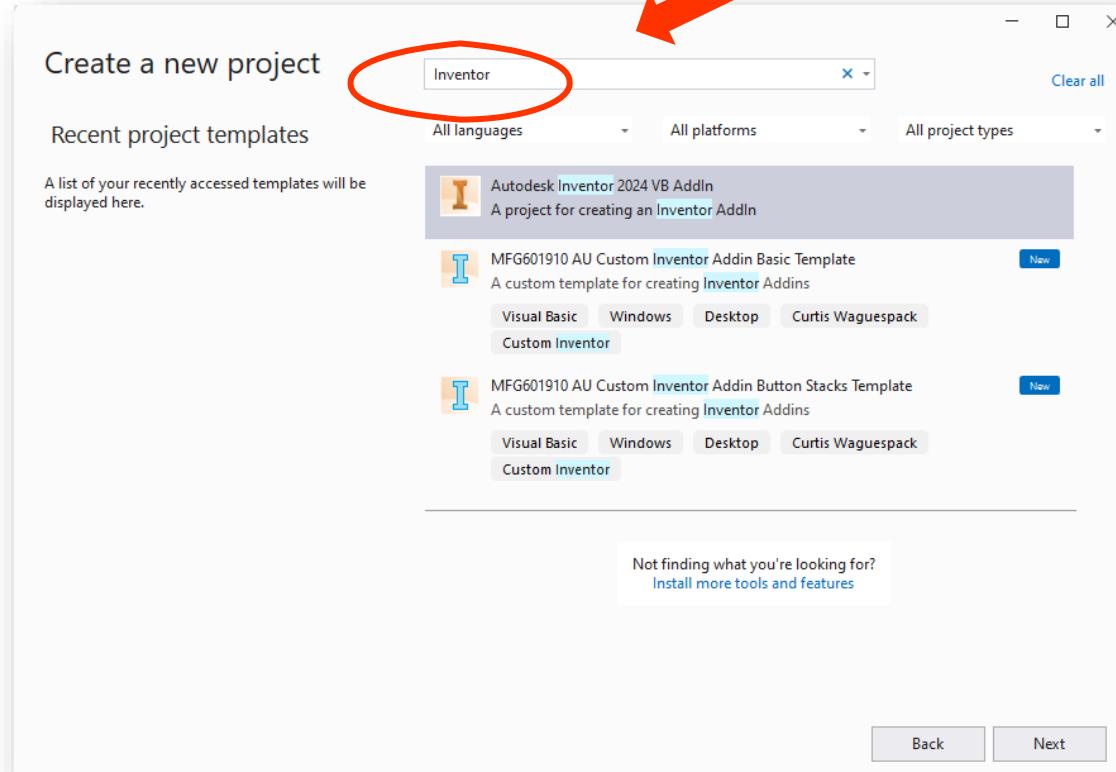




# Creating an Add-in

## 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

Create a new project

Recent project templates

A list of your recently accessed templates will be displayed here.

Search for templates (Alt+S)

All languages All platforms

Curtis Waguespack

MFG601910 AU Custom Inventor Addin Button Stacks Template  
A custom template for creating Inventor Addins  
Visual Basic Windows Desktop Curtis Waguespack  
Custom Inventor

MFG601910 AU Custom Inventor Addin Basic Template  
A custom template for creating Inventor Addins  
Visual Basic Windows Desktop Curtis Waguespack  
Custom Inventor

Not finding what you're looking for?  
[Install more tools and features](#)

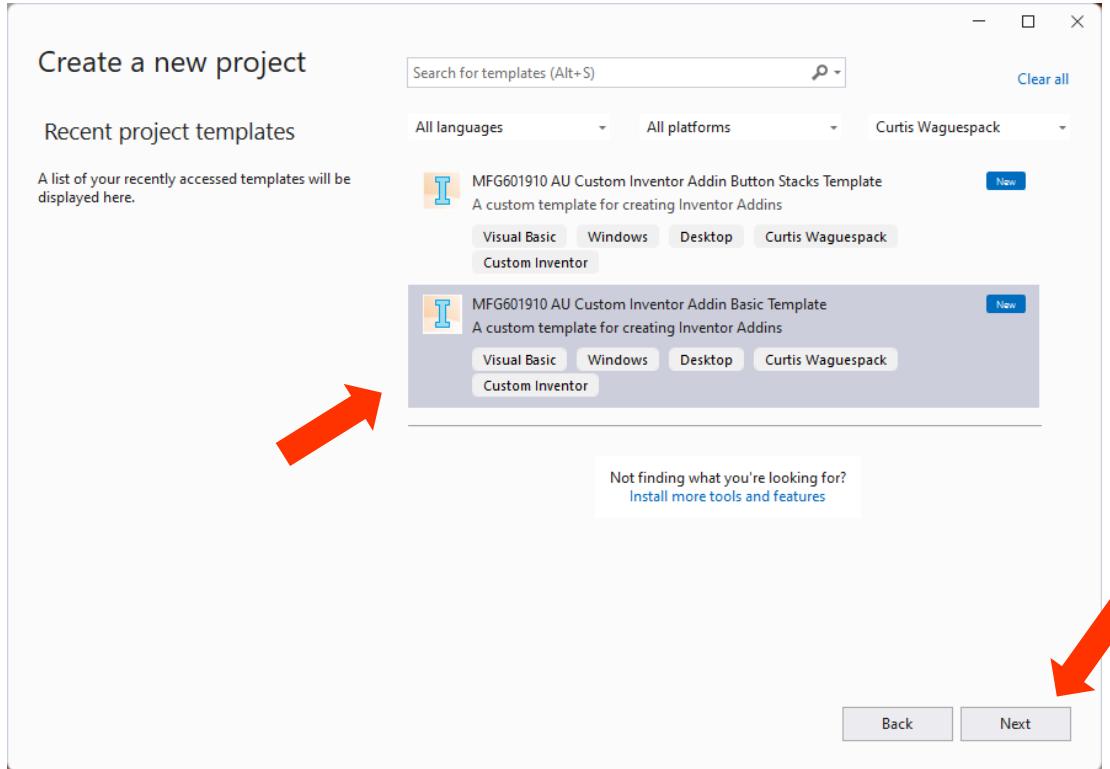
Back Next



# 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

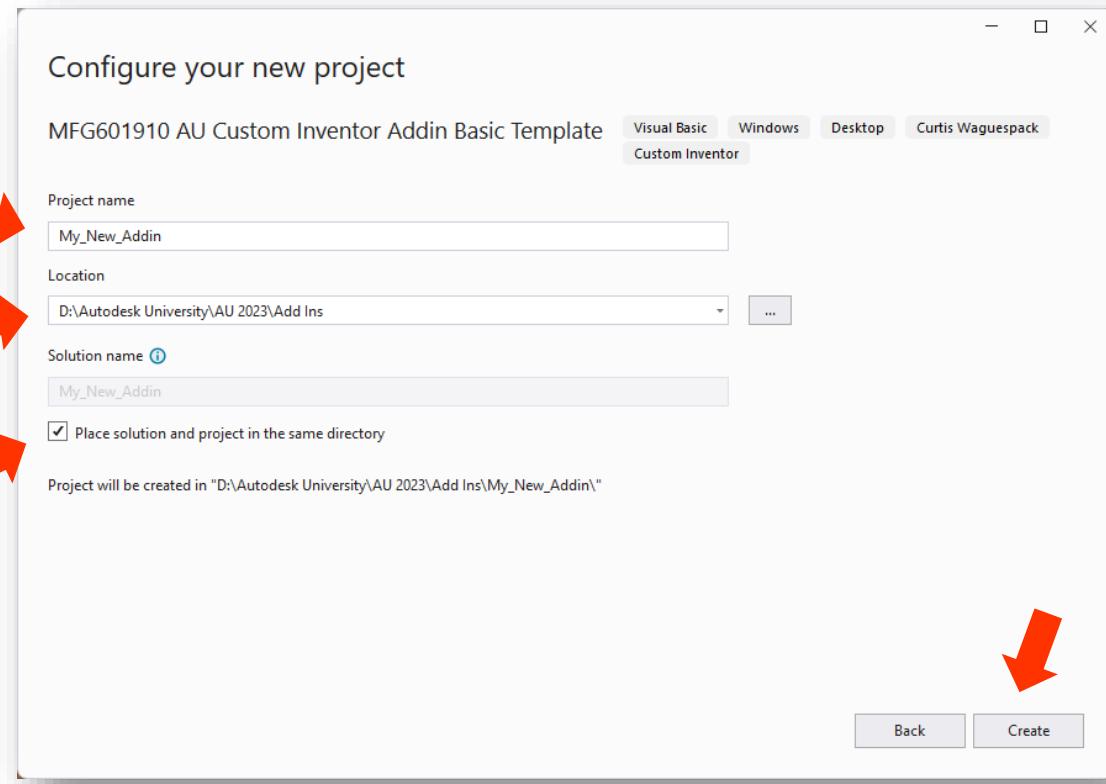
- Enter the project name

- 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

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

The screenshot shows the Microsoft Visual Studio interface. The ribbon bar at the top has a yellow highlight over the 'View' tab. Below the ribbon is the 'Solution Explorer' window, which displays a solution named 'My\_New\_Addin'. The solution contains several files and folders, including 'My Project', 'References', 'Command Tools' (with files 'API\_Help.vb', 'Detect\_Dark\_Light\_Theme.vb', and 'Welcome.vb'), 'Resources', 'Utility Tools', 'AssemblyInfo.vb', 'Autodesk.My\_New\_Addin.Inventor.addin', 'My\_New\_Addin.manifest', 'Readme.txt', and 'StandardAddInServer.vb'. A red arrow points from the woman's pointing hand towards the 'Solution Explorer' window.

# Creating an Add-in



The screenshot shows the Microsoft Visual Studio interface. In the top navigation bar, the 'File' tab is selected. The 'Solution Explorer' window is open, displaying a solution named 'My\_New\_Addin' containing several files and folders. A context menu is open over the solution node, with the 'Open Folder in File Explorer' option highlighted by a red rectangle. The 'Output' window at the bottom shows the status 'Ready'.

- 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 screenshot shows the Microsoft Visual Studio interface with the following details:

- Solution Explorer:** Displays the project structure for "My\_New\_Addin".
- Items in Solution:**
  - Folder: .vs (File folder)
  - Folder: bin (File folder)
  - Folder: Command Tools (File folder)
  - Folder: My Project (File folder)
  - Folder: obj (File folder)
  - Folder: Resources (File folder)
  - Folder: Utility Tools (File folder)
  - File: AssemblyInfo.vb (Visual Basic Source File)
  - File: Autodesk.My\_New\_Addin.Inventor.ad... (ADDIN File)
  - File: My\_New\_Addin.manifest (MANIFEST File)
  - File: My\_New\_Addin.sln (Visual Studio Solution)
  - File: My\_New\_Addin.vbproj (Visual Basic Project File)
  - File: My\_New\_Addin.vbproj.user (Per-User Project Options File)
  - File: Readme.txt (Text Document)
  - File: StandardAddInServer.vb (Visual Basic Source File)
- Error List:** Shows 0 errors.
- Output:** Shows the output of the build process.

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

# Creating an Add-in



The screenshot shows the Microsoft Visual Studio interface. The top menu bar includes File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help, and a Search bar. The toolbar below has icons for Undo, Redo, Cut, Copy, Paste, Find, Replace, and others. The status bar at the bottom shows "Build succeeded" and "Ready".

The main area displays the Solution Explorer, which lists one project named "My\_New\_Add-In". A context menu is open over the project node, with the "Build Solution" option highlighted by a red box and a cursor. Other options in the menu include Rebuild Solution, Clean Solution, Analyze and Code Cleanup, Configuration Manager..., Manage NuGet Packages for Solution..., Restore NuGet Packages, New Solution Explorer View, Add, Configure Startup Projects..., Create Git Repository..., Paste, Rename, Copy Full Path, Open Folder in File Explorer, Open in Terminal, Save As Solution Filter, Hide Unloaded Projects, and Properties.

The Output window below shows the build logs:

```
Show output from: Build
1> INFO: Could not find files for the given pattern(s).
1> The system cannot find the path specified.
1> 'mt.exe' is not recognized as an internal or external command,
1> operable program or batch file.
1> D:\Autodesk University\AU 2023\Add Ins\My_New_Add-In\bin\Debug\My_New_
1> 1 File(s) copied
1> D:\Autodesk University\AU 2023\Add Ins\My_New_Add-In\Autodesk.My_New_Add-In.Inventor.addin
1> 1 File(s) copied
===== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped =====
===== Build started at 4:25 PM and took 00.680 seconds =====
```

The Error List tab is selected in the Output window.

- The solution is shown in the Solution Explorer



# Creating an Add-in



The screenshot shows the Microsoft Visual Studio interface. A yellow sticky note on the left side of the menu bar says: "Tip: you can turn the Output window on/off using the View tab". The menu bar is open, and the "Output" option is highlighted with a blue selection bar. The main window shows the Solution Explorer with a project named "My\_New\_Addin". The Output window at the bottom displays the build logs:

```
Show output from: Build
1> INFO: Could not find files for the given pattern(s).
1> The system cannot find the path specified.
1> 'mt.exe' is not recognized as an internal or external command,
1> operable program or batch file.
1> D:\Autodesk University\AU 2023\Add Ins\My_New_AddIn\bin\Debug\My_New_AddIn.dll
1> 1 File(s) copied
1> D:\Autodesk University\AU 2023\Add Ins\My_New_AddIn\Autodesk.My_New_AddIn.Inventor.addin
1> 1 File(s) copied
===== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped =====
===== Total 1 item(s) and took 00.723 seconds =====
|
```

A yellow sticky note on the right side of the screen says: "Note the output logging indicates the solution build status." The "Output" tab is selected in the Output window.

- The Build results are shown in the Output window

# Overall Process



# Visual Studio



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



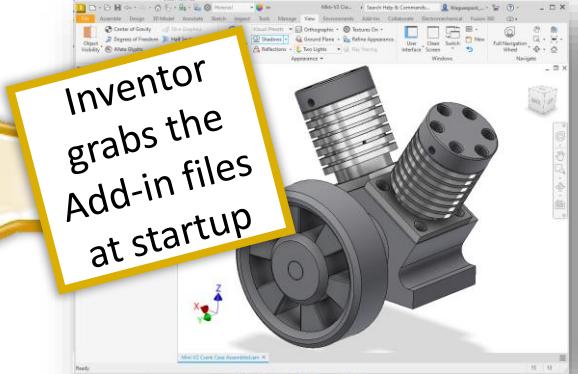
# Autodesk Application Plugins Folder



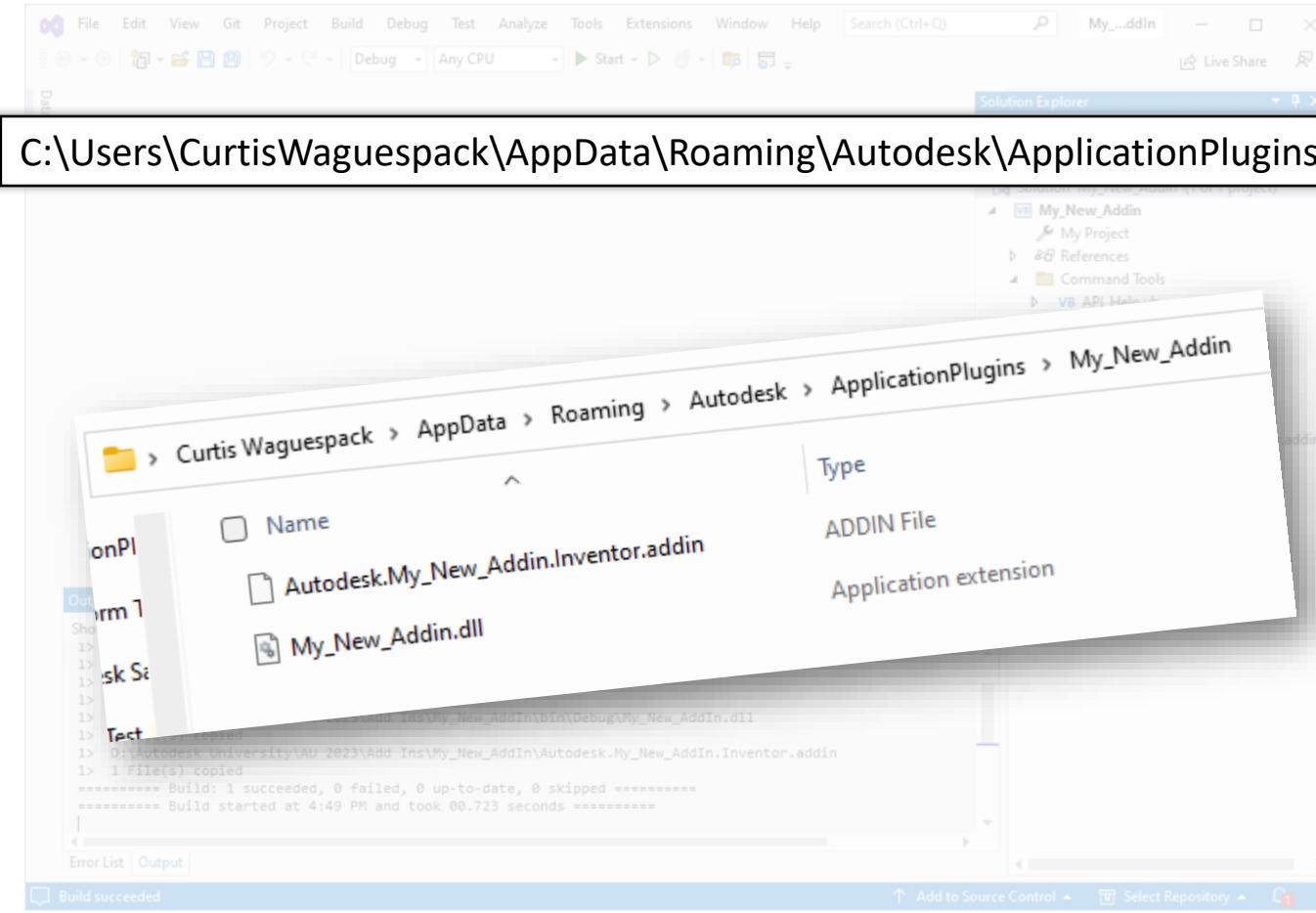
# Review



# Autodesk Inventor



# Creating an Add-in



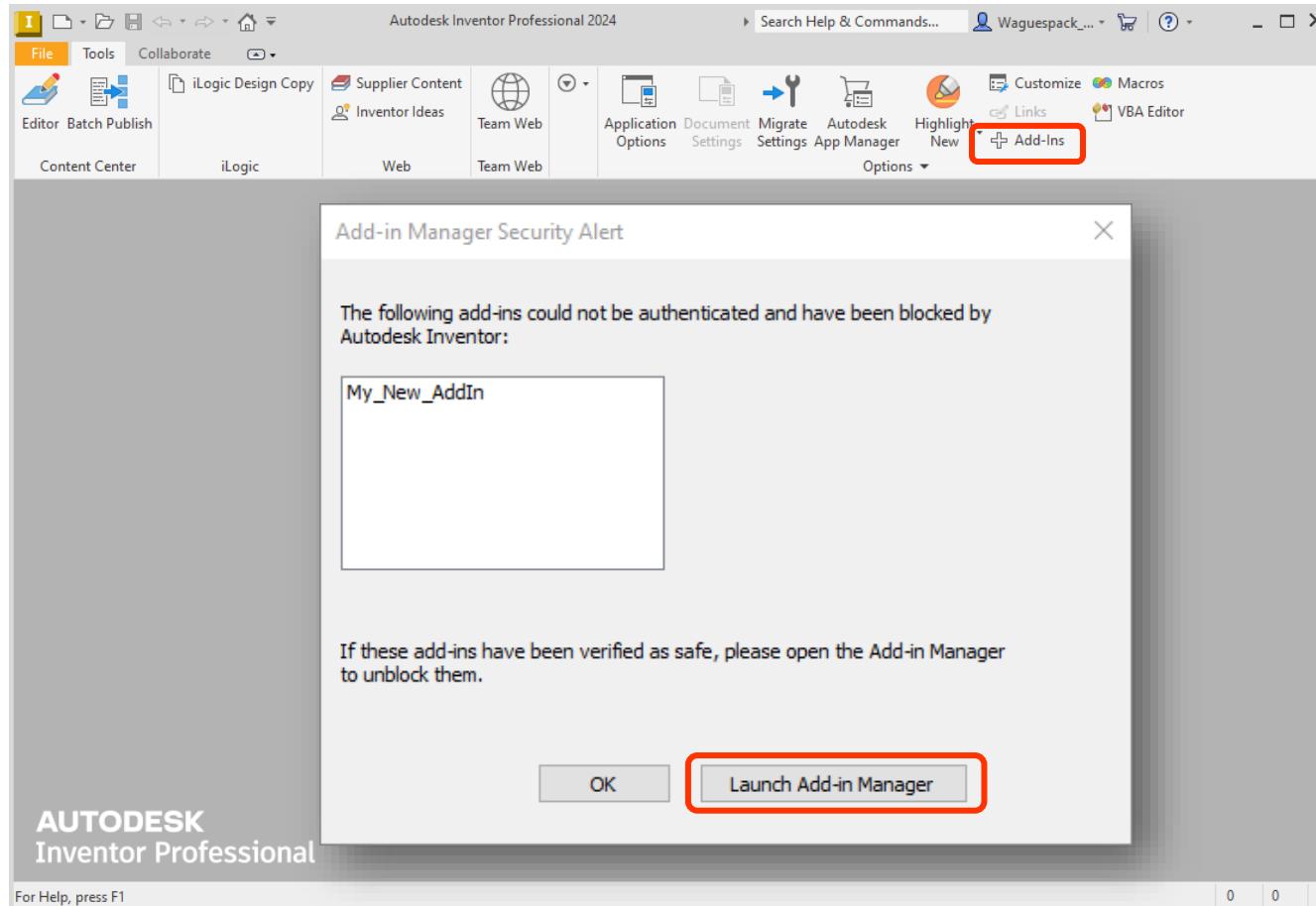
- Browse to the ApplicationPluggins folder and observe the files created during the Build

# Loading the add-in

Working in Inventor

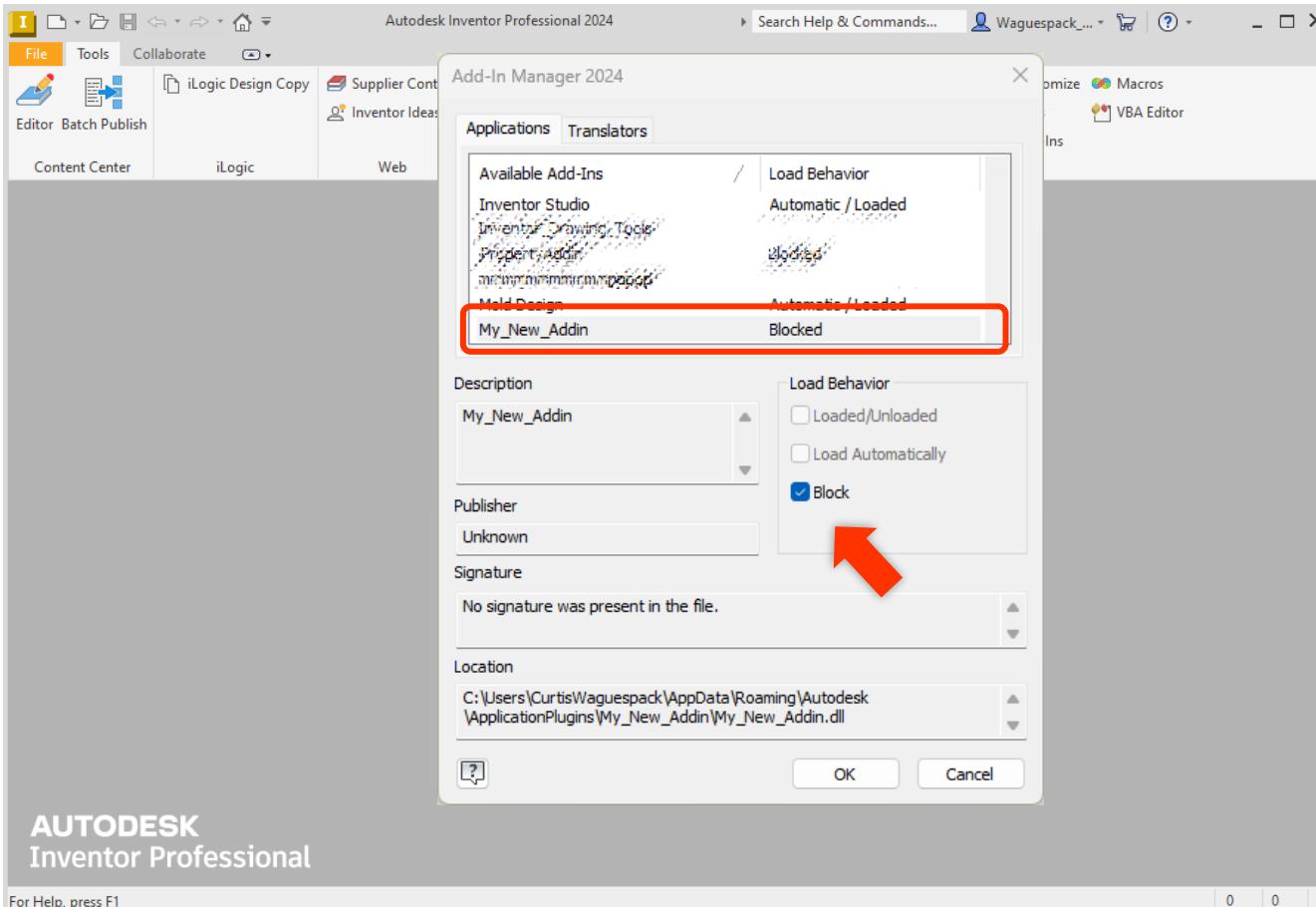


# Loading an Add-in



- 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 go 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



The screenshot shows the Autodesk Inventor Professional 2024 interface with the 'Add-In Manager 2024' dialog box open. The 'Available Add-Ins' list includes 'Inventor Studio', 'Inventor Drawing Tools', 'Property Manager', 'Mold Design', and 'My\_New\_Addin'. The 'Load Behavior' for 'My\_New\_Addin' is set to 'Automatic / Loaded'. In the 'Load Behavior' section of the dialog, three checkboxes are visible: 'Loaded/Unloaded' (checked), 'Load Automatically' (checked), and 'Block' (unchecked). A red box highlights the 'Load Behavior' section. The 'Description' field shows 'My\_New\_Addin', 'Publisher' is 'Unknown', and 'Signature' indicates 'No signature was present in the file.' The 'Location' is listed as 'C:\Users\CurtsWaguespack\AppData\Roaming\Autodesk\ApplicationPlugins\My\_New\_Addin\My\_New\_Addin.dll'. At the bottom are 'OK' and 'Cancel' buttons.



**AUTODESK**  
Inventor Professional

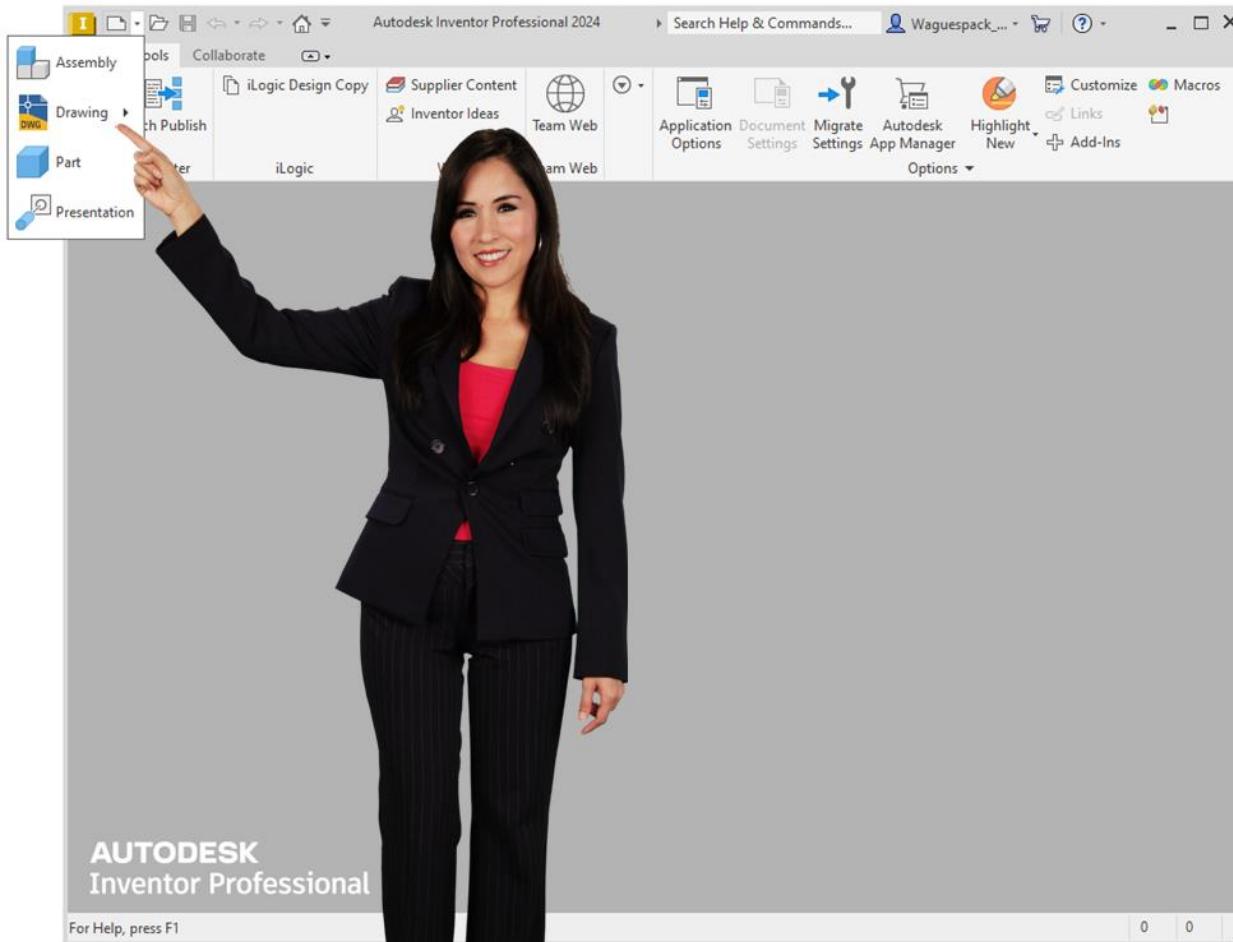
For Help, press F1

# Using the add-in

Working in Inventor

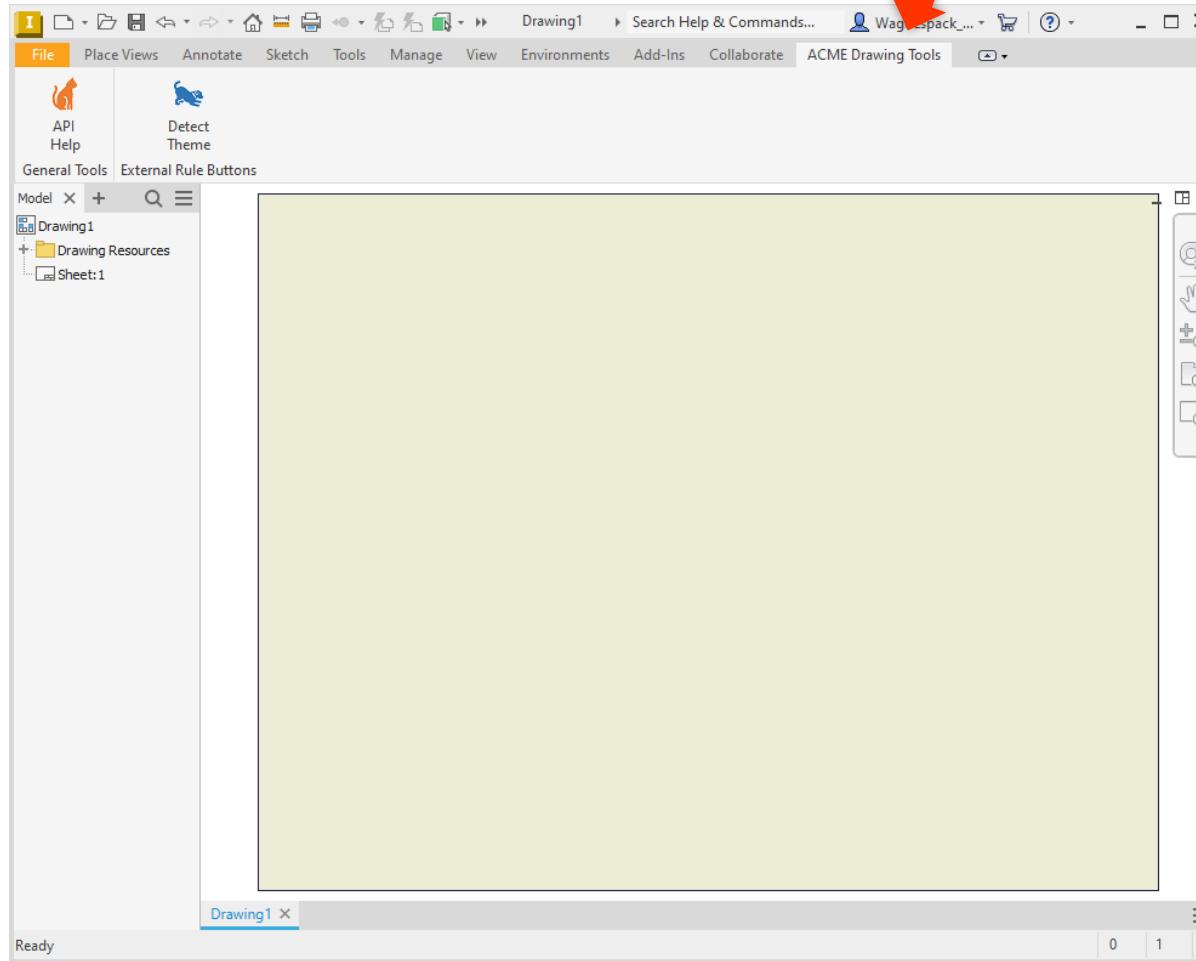


# Using the Add-in



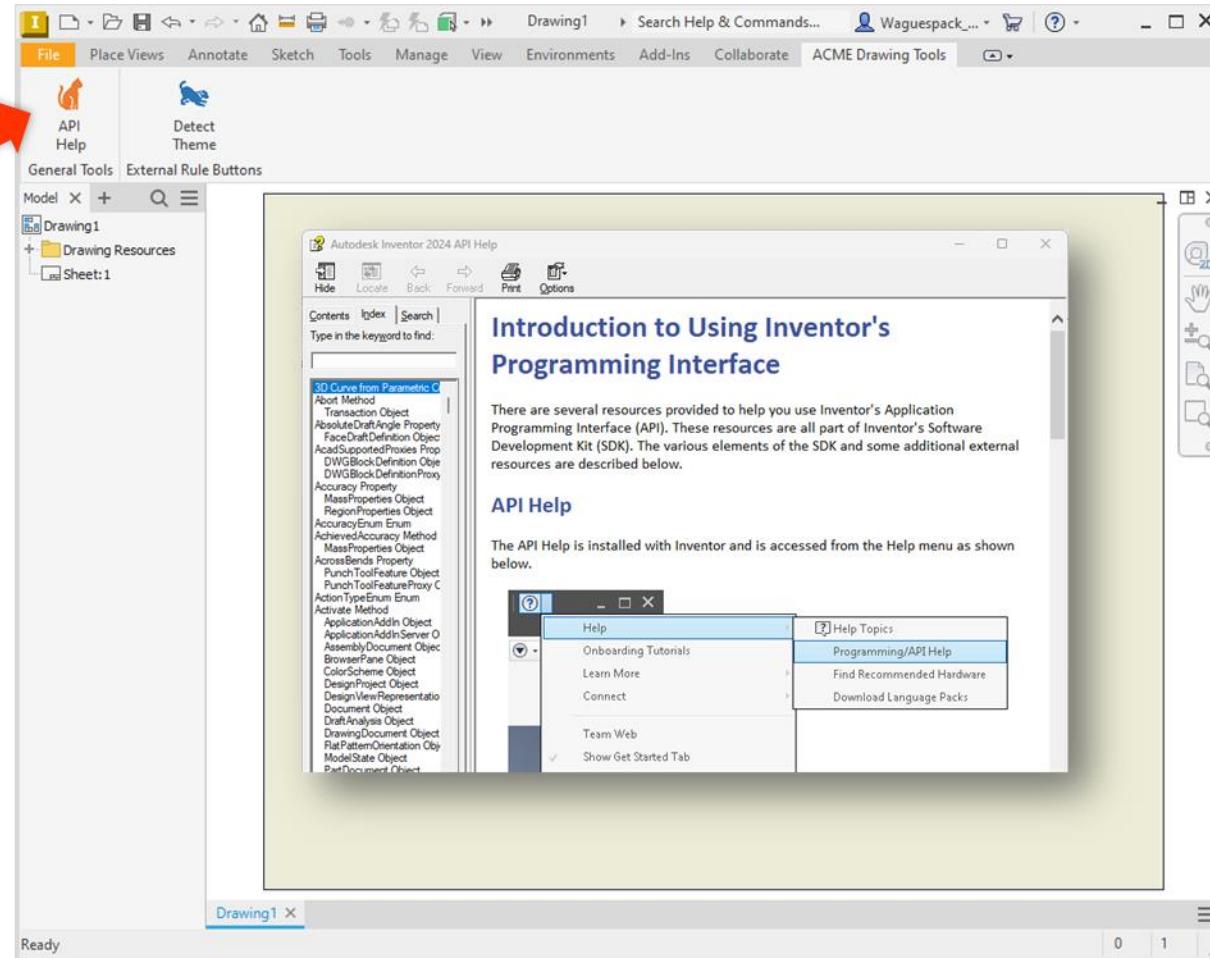
- Create a new drawing file from a template

# Using the Add-in



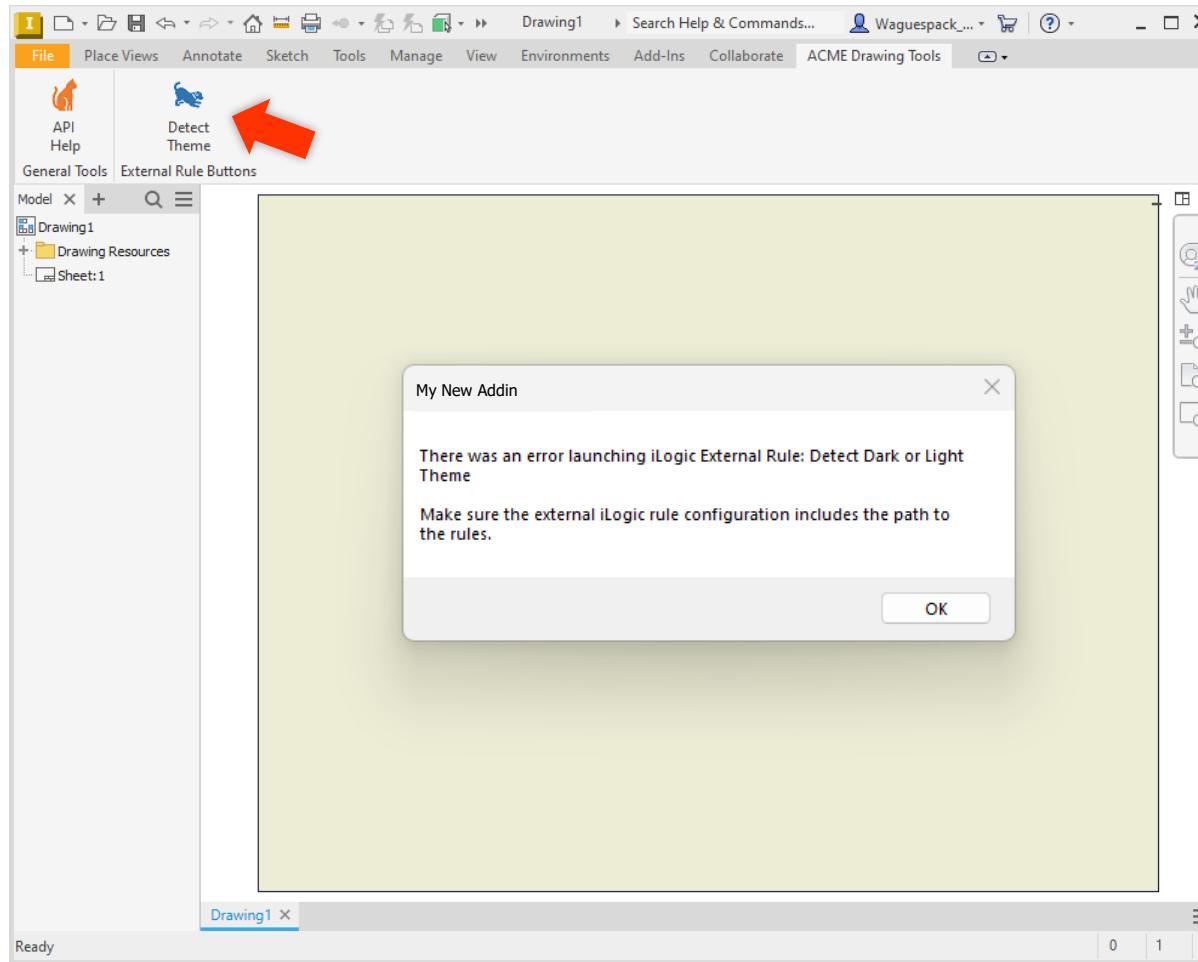
- Switch to the **ACME Drawing Tool** tab
- Note the 2 buttons found on this tab

# Using the Add-in



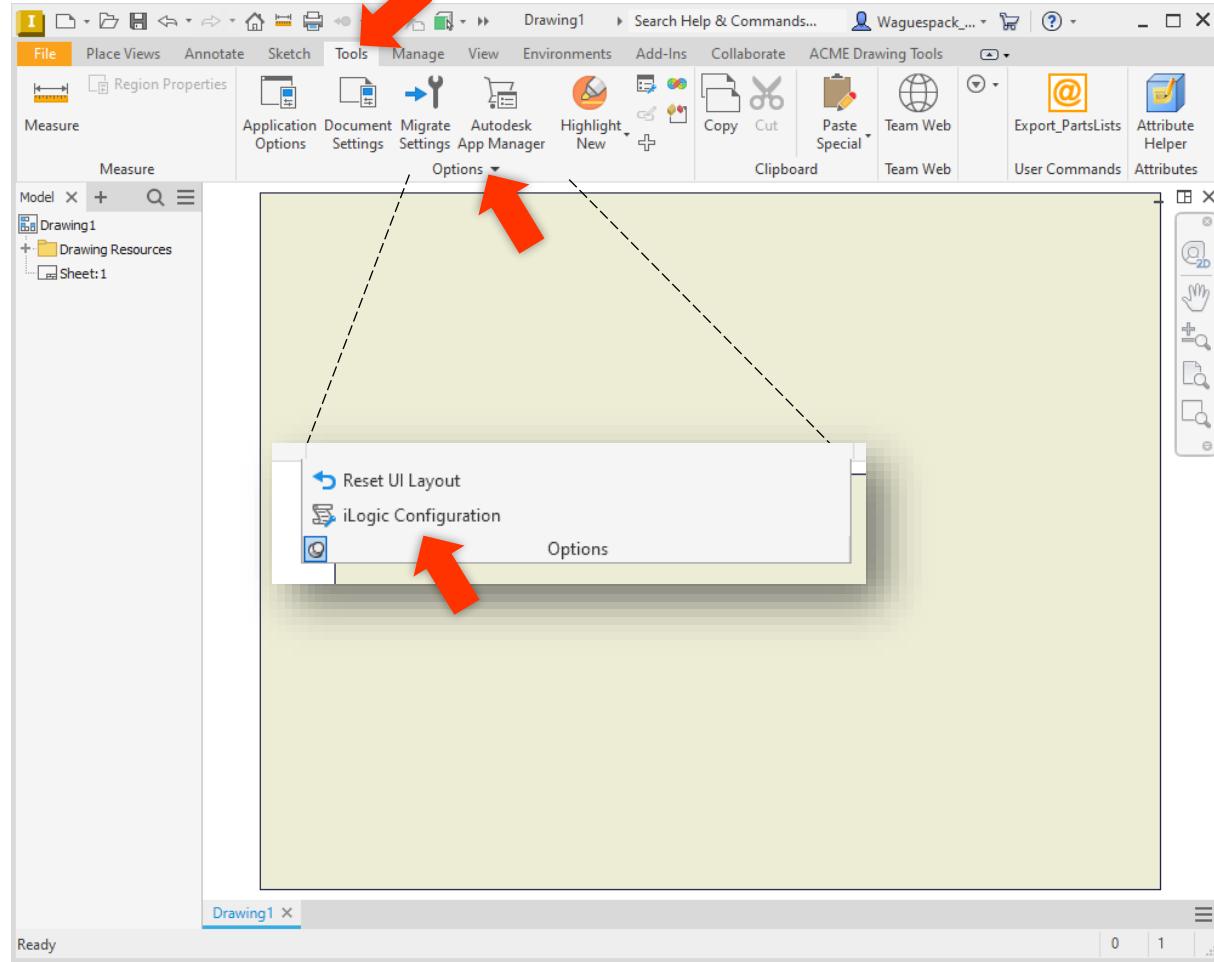
- Click the button called **API Help**
- This will open the Inventor API Help in a separate window
- This was done with code written in the add-in
- You can close the window if you like

# Using the Add-in



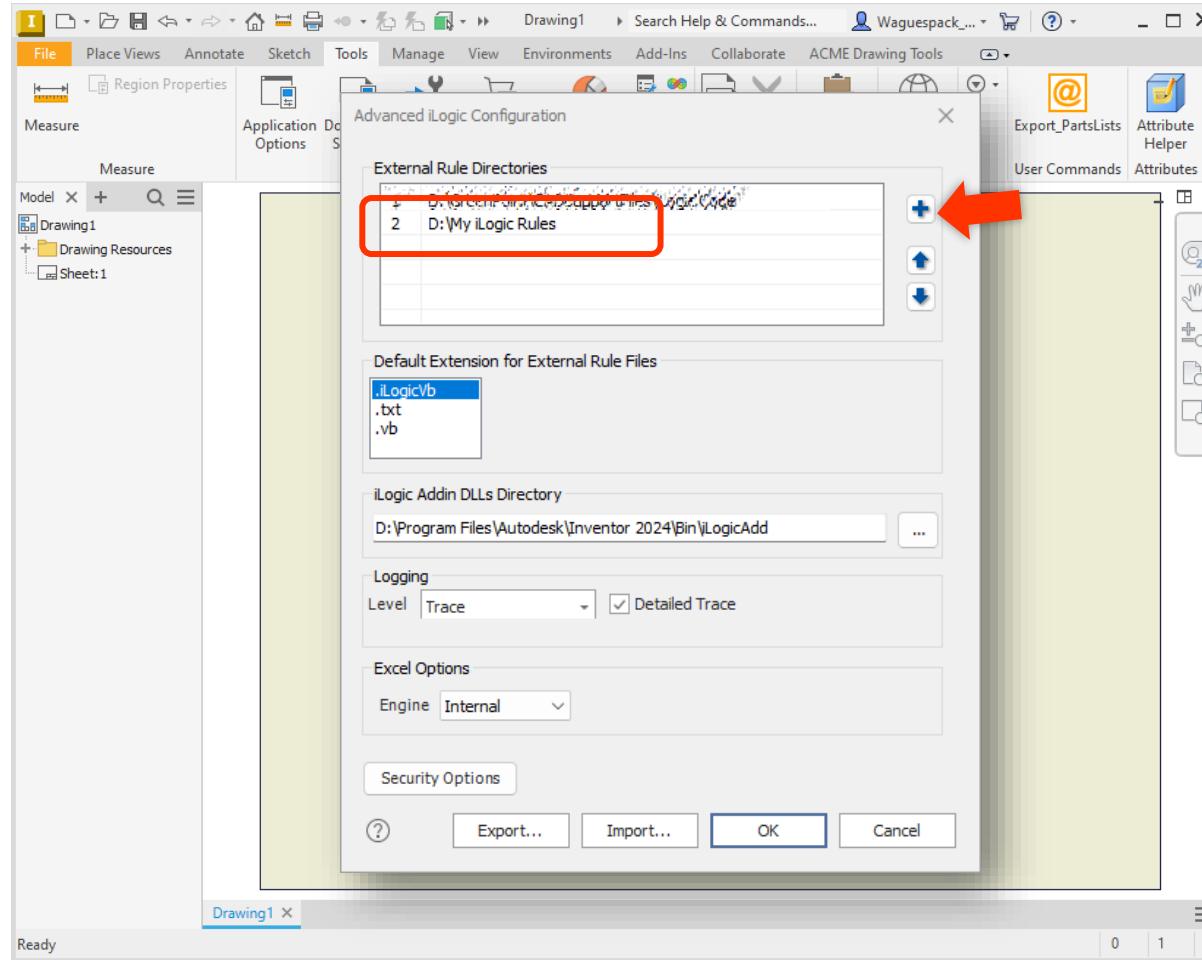
- Click the button called **API Help**
- This will open the Inventor API Help in a separate window
- This was done with code written in the add-in
- You can close the window if you like

# Using the Add-in



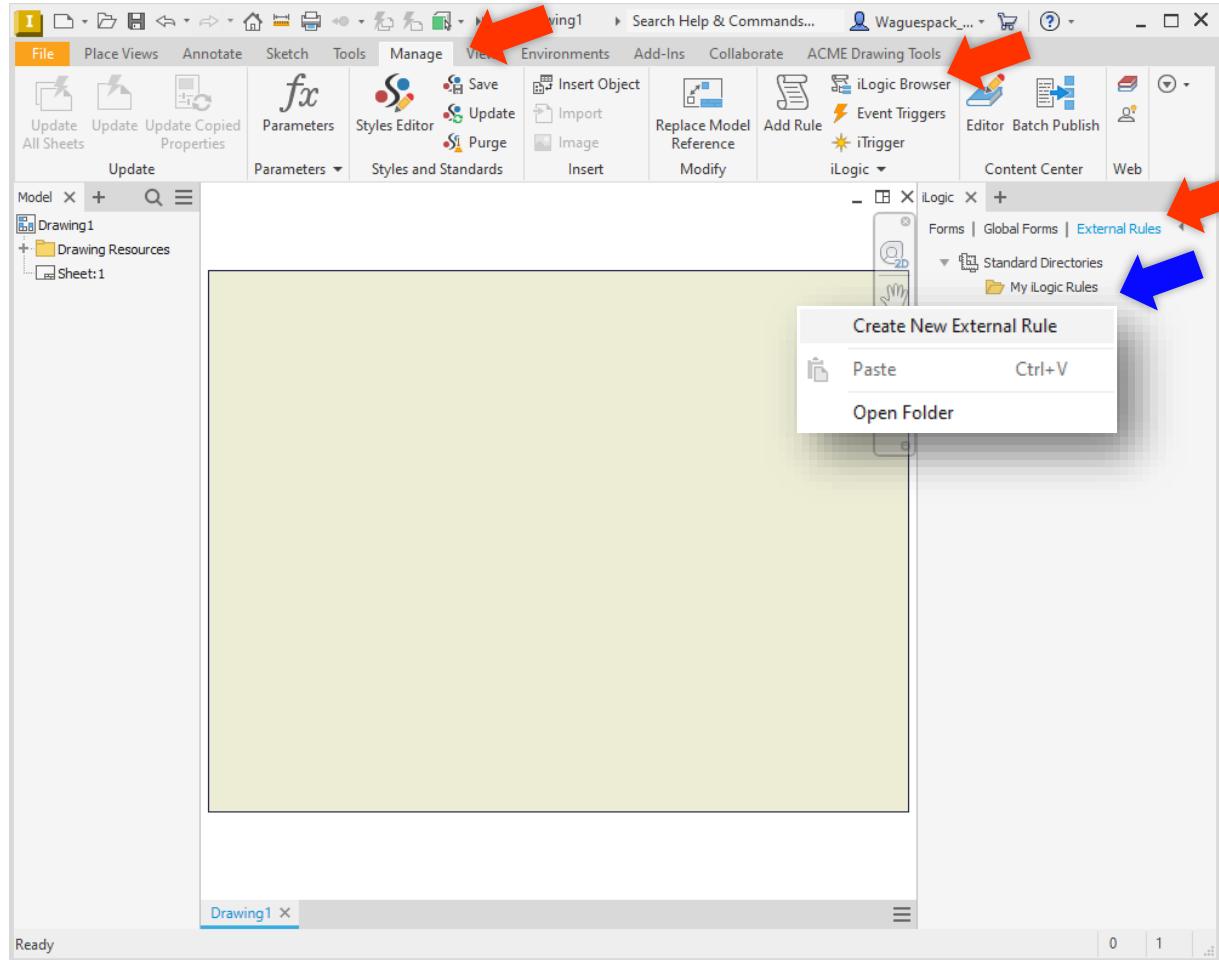
- Switch to the **Tools** tab and click the **Options** button
- Click the **iLogic Configuration** button as shown

# Using the Add-in



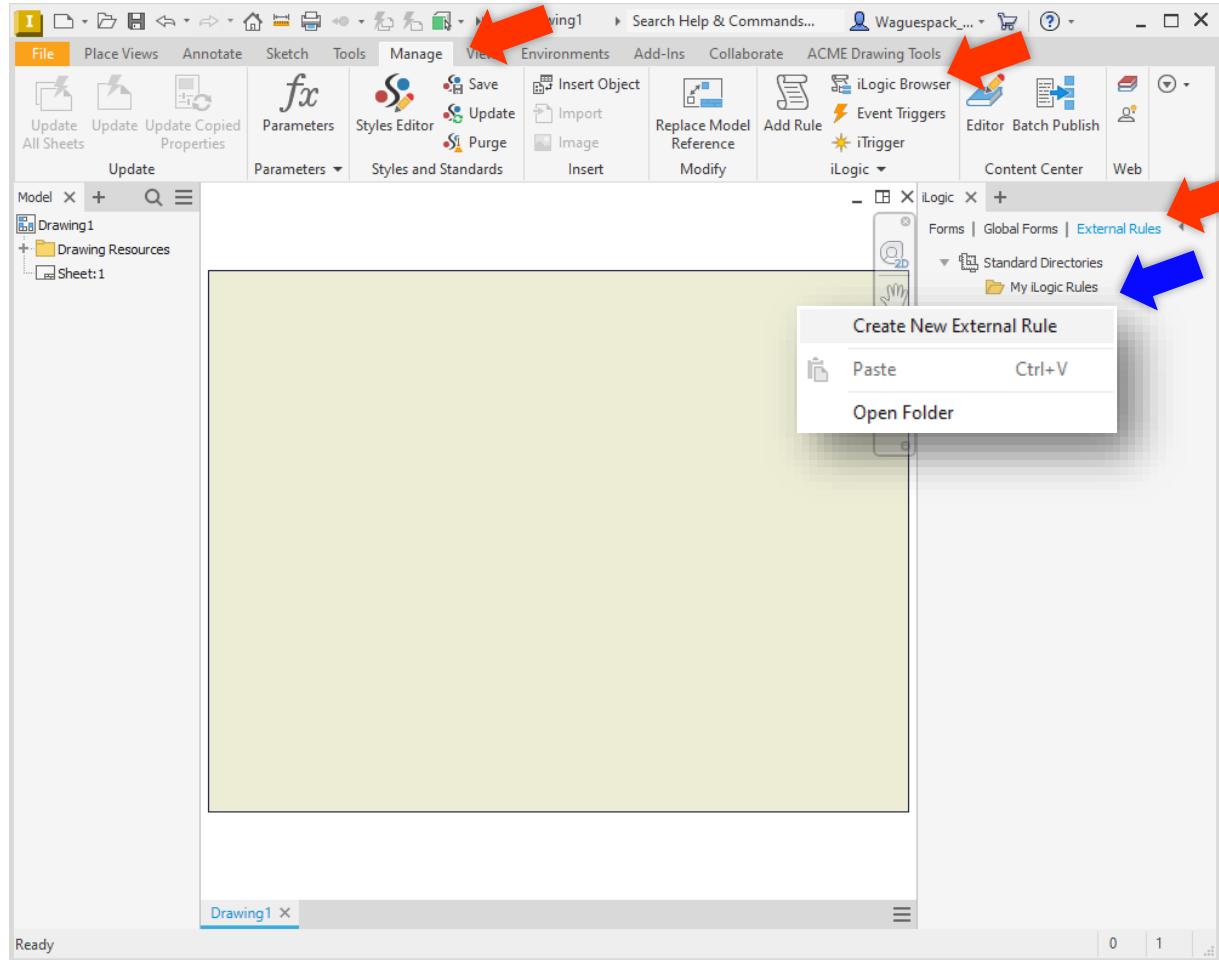
- If you have an External Rule directory already specified you can just use it
- If not use the + button to add a new path of your choice
- Click the **OK** button when done

# 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

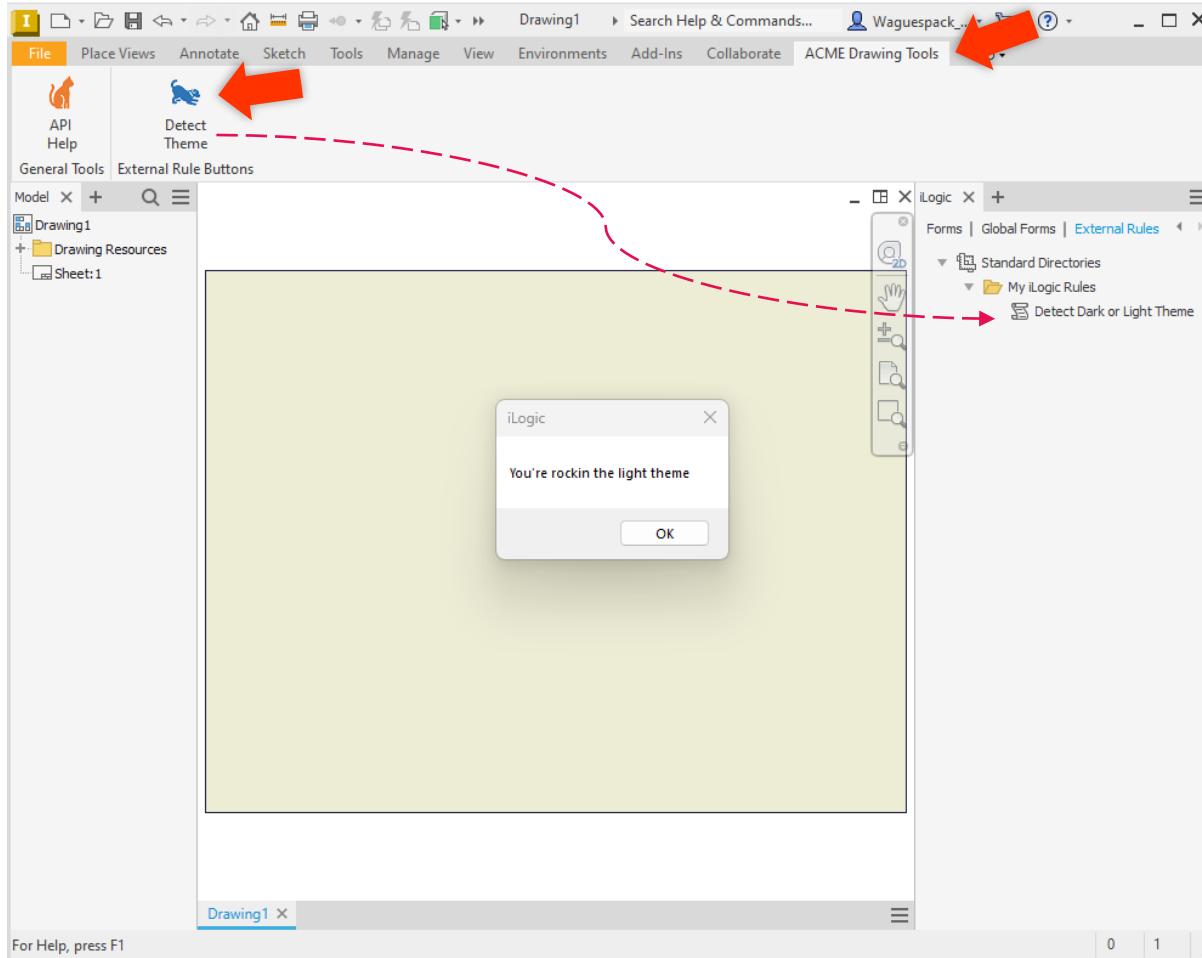
A screenshot of the ACME Drawing Tools software interface. The top menu bar includes File, Place Views, Annotate, Sketch, Tools, Manage, View, Environments, Add-Ins, Collaborate, ACME Drawing Tools, and a user profile. The Manage tab is selected. Below the menu is a toolbar with icons for Update All Sheets, Update Copied Properties, Parameters, Styles Editor, Save, Import, Replace Model Reference, Add Rule, Event Triggers, iTrigger, Editor, Batch Publish, and a Help icon. A central window titled "Edit Rule: Detect Dark or Light Theme" shows the iLogic Editor. The left sidebar lists categories like System, Custom, Parameters, Features, Components (classic), etc. The main editor area has tabs for Model, Options, Search and Replace, and Wizards. Under Model, "Drawing1" is selected, showing "Sheet:1". The iLogic code editor contains the following VBA-like script:

```
1 Dim activeTheme As String
2 activeTheme = ThisApplication.ThemeManager.ActiveTheme.Name
3
4 If activeTheme = "DarkTheme" Then
5     MsgBox("You're rockin the dark theme", "iLogic")
6 ElseIf activeTheme = "LightTheme" Then
7     MsgBox("You're rockin the light theme", "iLogic")
8 End If
9
```

The bottom of the editor shows Log Level (Trace), Detailed Trace checked, and buttons for Save, Save & Run, and Cancel. The status bar at the bottom indicates "Ready".

- 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



A screenshot of the SolidWorks interface demonstrating the use of the ACME Add-in across different environments. The top navigation bar shows tabs for File, Assemble, Design, 3D Model, Annotate, Sketch, and Inspect, along with a custom tab labeled 'ACME Assembly Tools'. A red arrow points to the user profile icon in the top right of the main window. Below, three separate windows are shown: 1) Assembly environment with tabs for File, 3D Model, Sketch, Annotate, Inspect, Tools, Fusion 360, and ACME Part Tools. 2) Part environment with tabs for File, Place Views, Annotate, Sketch, Tools, Manage, View, Environments, Add-Ins, Collaborate, and ACME Drawing Tools. 3) Drawing environment with tabs for File, Place Views, Annotate, Sketch, Tools, Manage, View, Environments, Add-Ins, Collaborate, and ACME Drawing Tools. Red arrows point to the user profile icons in the top right of each of these three windows. The bottom of the interface shows General Tools and External Rule Buttons sections.

- 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

# 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

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help

Solution Explorer Ctrl+Alt+L

Git Changes Ctrl+0, Ctrl+G

Git Repository Ctrl+0, Ctrl+R

Team Explorer Ctrl+\, Ctrl+M

Server Explorer Ctrl+Alt+S

Data Lake Analytics Explorer

SQL Server Object Explorer Ctrl+\, Ctrl+S

Test Explorer Ctrl+E, T

Any CPU Start Live Share

Solution Explorer

Search Solution Explorer (Ctrl+.)

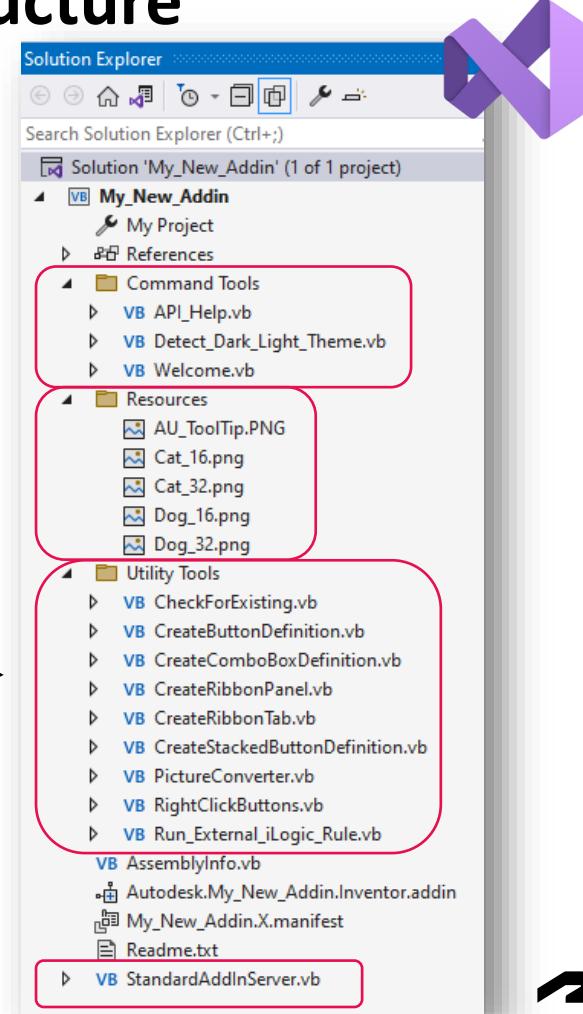
Solution 'My\_New\_Addin' (1 of 1 project)

- My\_New\_Addin
  - My Project
  - References
  - Command Tools
    - API.Help.vb
    - Detect\_Dark\_Light\_Theme.vb
    - Welcome.vb
  - Resources
    - AU\_ToolTip.PNG
    - Cat\_16.png
    - Cat\_32.png
    - Dog\_16.png
    - Dog\_32.png
  - Utility Tools
    - CheckForExisting.vb
    - CreateButtonDefinition.vb
    - CreateComboBoxDefinition.vb
    - CreateRibbonPanel.vb
    - CreateRibbonTab.vb
    - CreateStackedButtonDefinition.vb
    - PictureConverter.vb
    - RightClickButtons.vb
    - Run\_External\_iLogic\_Rule.vb
  - AssemblyInfo.vb
  - Autodesk.My\_New\_Addin.Inventor.addin
  - My\_New\_Addin.X.manifest
  - Readme.txt
  - StandardAddInServer.vb

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



# Understanding the Add-in Template Structure



Command Tool Modules →

Resources →

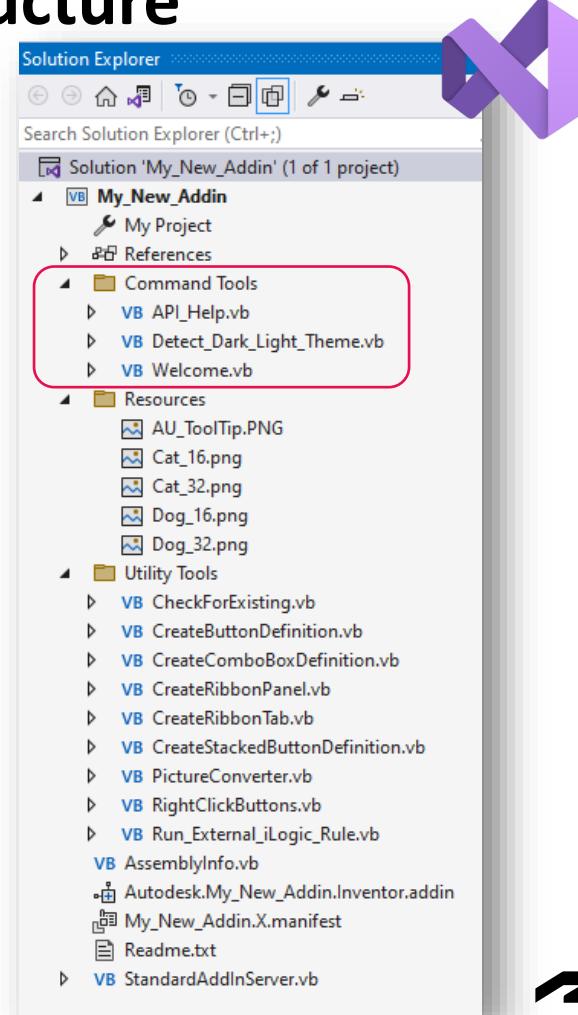
Utility Modules →

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* ✘ X
VB My_New_Addin
Imports Inventor
Module API_Help
    ''' <summary> Creates a button Button Definition Define the label, icons, tool t ...
    'This function is where the button is defined
    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, useL:
        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* ✘ X
VB My_New_Addin
Imports Inventor
Module API_Help
    'summary> Creates a button Button Definition Define the label, icons, tool t ...
    'This function is where the button is defined
    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 " & vbCrLf & "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

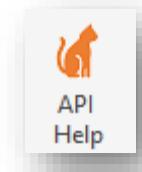
# Understanding the Add-in Template Structure



```
API_Help.vb* ✘ X
VB My_New_Addin API_Help CreateButton
Imports Inventor
Module API_Help
    ''' <summary> Creates a button Button Definition Define the label, icons, tool t ...
    'This function is where the button is defined
    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 " & vbCrLf & "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
```



Specify the icons



# Understanding the Add-in Template Structure



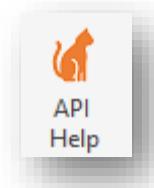
```
API_Help.vb* ✘ X
VB My_New_Addin API_Help CreateButton
Imports Inventor
Module API_Help
    'summary> Creates a button Button Definition Define the label, icons, tool t ...
    'This function is where the button is defined
    Function CreateButton(environment As String, CustomDrawingTab As RibbonTab, 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 " & vbCrLf & "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
```

Specify the Button  
Display Name



# Understanding the Add-in Template Structure



API\_Help.vb\* ✘ X

VB My\_New\_Addin API\_Help CreateButton

```
Imports Inventor

Module API_Help
    'summary> Creates a button Button Definition Define the label, icons, tool t ...
    'This function is where the button is defined
    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 " & vbCrLf & "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
```

Define the Tool Tip

The screenshot shows the Autodesk Inventor 2024 API Help window. The title bar says "Autodesk Inventor 2024 API Help". The main content area displays the "Introduction to Using Inventor's Programming Interface" page. The page includes a table of contents on the left and a detailed description of the API interface on the right. A red box highlights the "Press F1 for more help" link at the bottom of the page.

# Understanding the Add-in Template Structure



API\_Help.vb\* ✘ X

VB My\_New\_Addin API\_Help CreateButton

```
Imports Inventor
Module API_Help
    ''' <summary> Creates a button Button Definition Define the label, icons, tool t ...
    'This function is where the button is defined
    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 " & vbCrLf & "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
    End Function

    'Pass the information to the utility module
    Sub 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 Sub
End Module
```

# Understanding the Add-in Template Structure



API\_Help.vb\* ✘ X

VB My\_New\_Addin API\_Help

```
Imports Inventor
Module API_Help
    'This function is where the button is defined
    Function CreateButton(environment As String, CustomDrawingTab As RibbonTab, ribbonP
        useLargeIcon As Boolean, isInButtonStack As Boolean) As Button
        'get the images to use for the button
        Dim largeIcon As IPictureDisp = PictureDispConverter.ToIPictureDisp(My.Resource
        Dim standardIcon As IPictureDisp = PictureDispConverter.ToIPictureDisp(My.Resou
        Dim toolTipImage As IPictureDisp = Nothing

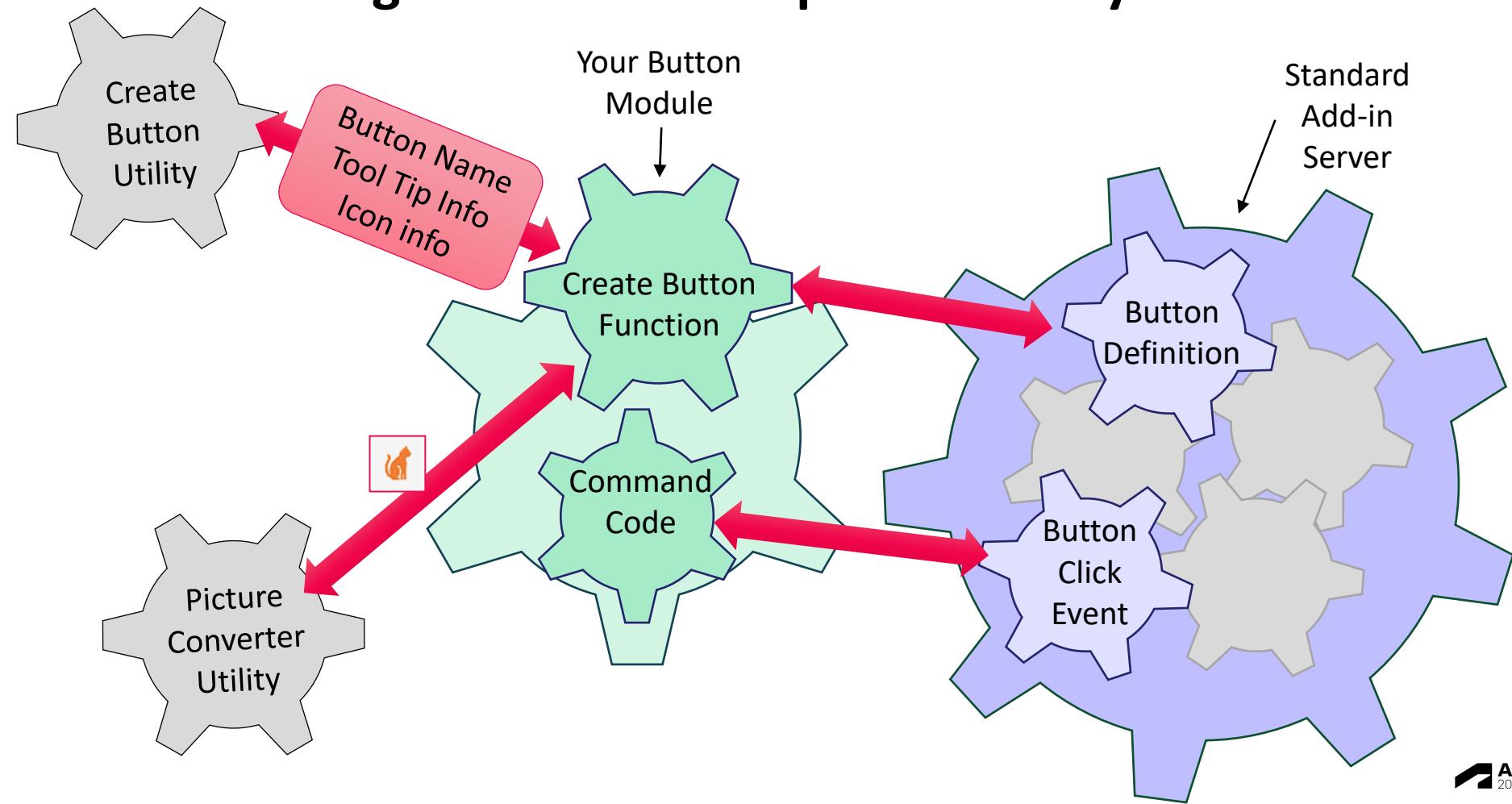
        'This is the code that does the real work when your command is executed.
        Sub RunCommandCode()
            Dim Version = g_inventorApplication.SoftwareVersion.DisplayVersion
            Dim Array = Split(Version, ".")
            Version = Array(0)
            Dim Build = g_inventorApplication.SoftwareVersion.Major

            Dim Filename = "C:\Users\Public\Documents\Autodesk\Inventor " &
            Version & "\Local Help\ADMAPI_" & Build & "_0.chm"

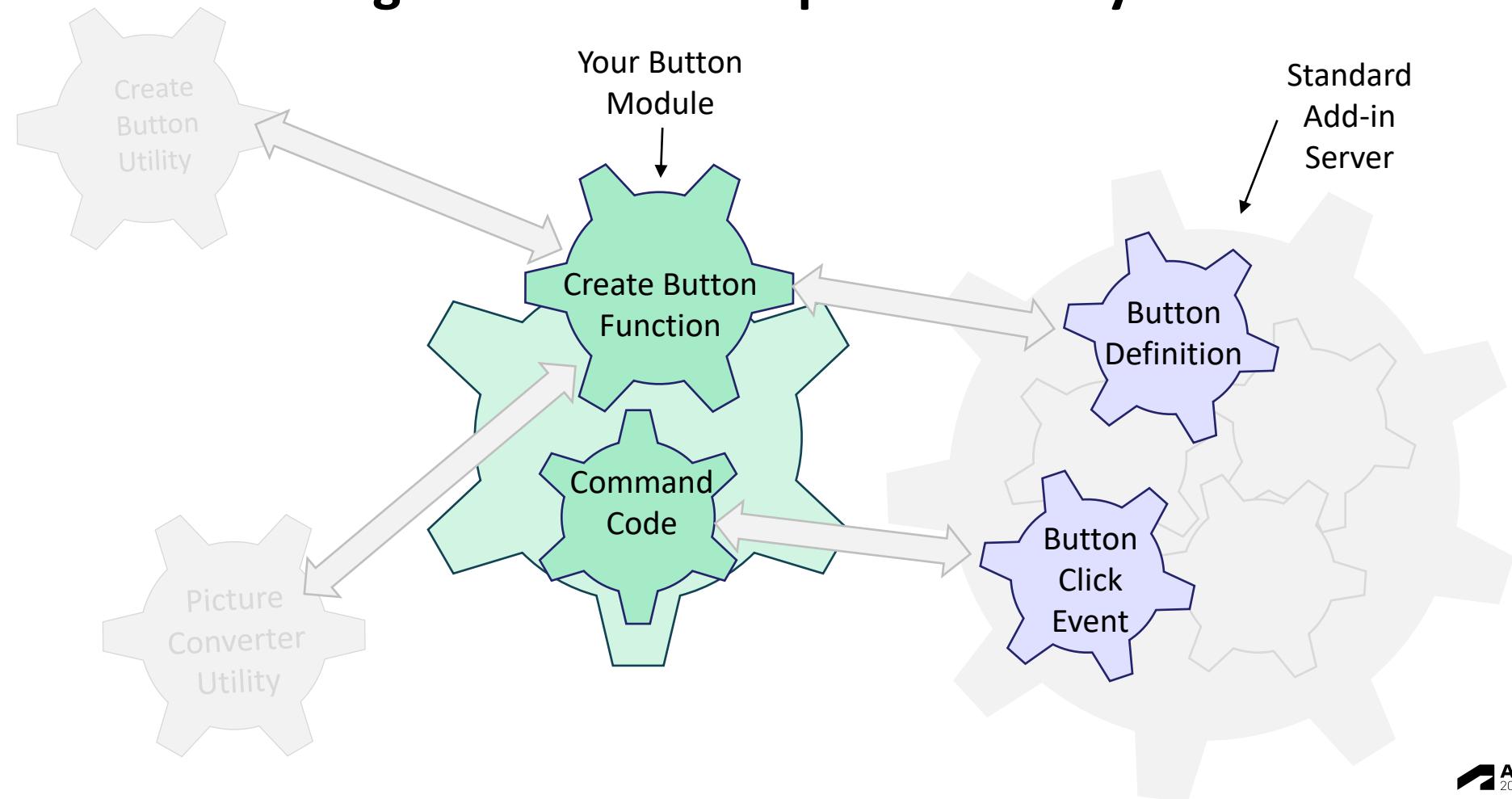
            If System.IO.File.Exists(Filename) = True Then
                Process.Start(Filename)
            Else
                MsgBox("File Does Not Exist" & vbCrLf & Filename)
            End If
        End Sub
    End Module
```

- 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 Visually

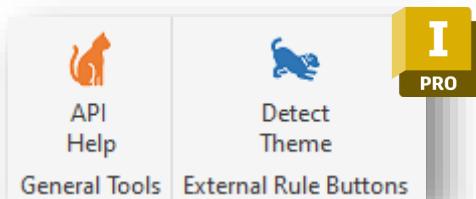
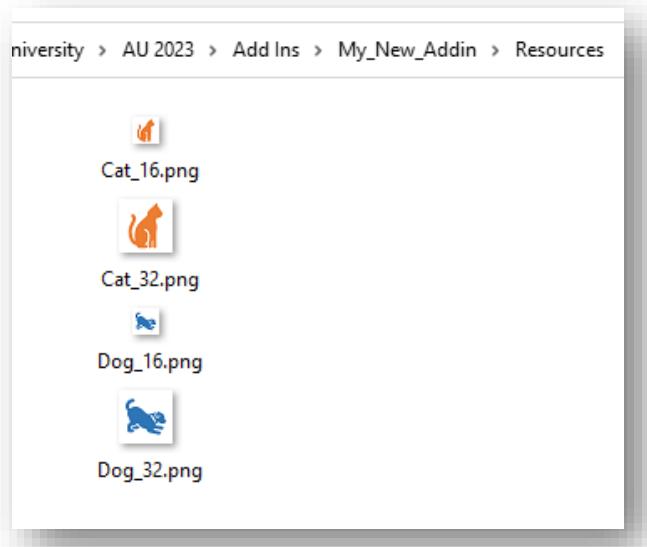


# Understanding the Add-in Template Visually

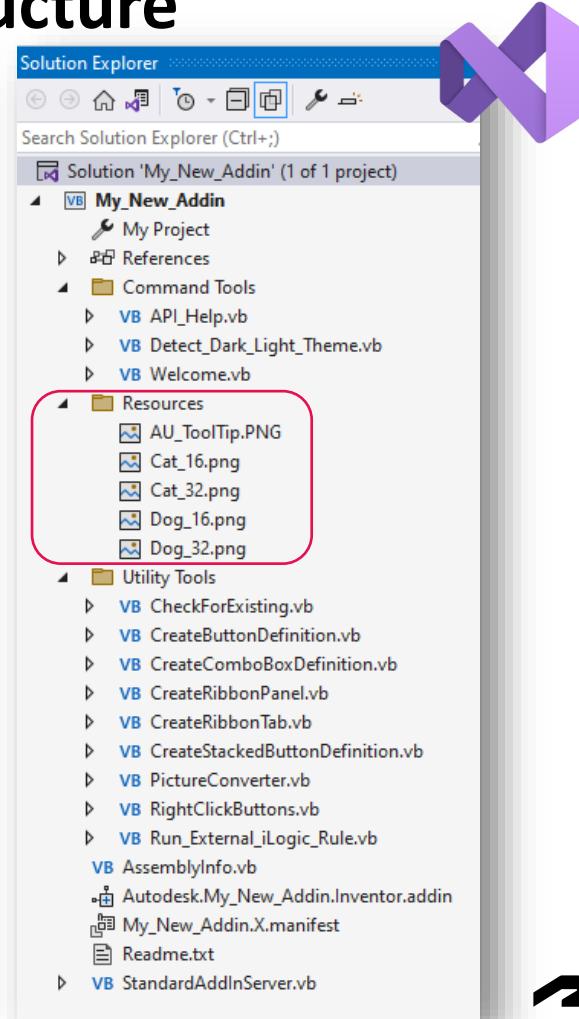


# Understanding the Add-in Template Structure

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



Resources →



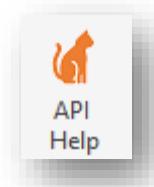
# Adding a button image



```
API_Help.vb* ✘ X
VB My_New_Addin API_Help CreateButton
Imports Inventor
Module API_Help
    ''' <summary> Creates a button Button Definition Define the label, icons, tool t ...
    'This function is where the button is defined
    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 " & vbCrLf & "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
```



Specify the icons



# Adding a button image



Screenshot of the Visual Studio IDE showing the Project menu open. A red arrow points to the 'Project' option in the menu bar. Another red arrow points to the 'My\_New\_Addin Properties' item in the bottom-left corner of the menu.

```
API File Edit View Git Project Build Debug
VB M
+ Add Form (Windows Forms)...
+ Add User Control (Windows Forms)...
+ Add Component...
+ Add Module...
+ Add Class...
+ Add New Data Source...
□ Add New Item... Ctrl+Shift+A
□ Add Existing Item... Shift+Alt+A
Exclude From Project
□ Show All Files
Add Reference...
Add Service Reference...
Connected Services
Add Analyzer...
⚙ Configure Startup Projects...
Set as Startup Project
Export Template...
🔧 My_New_Addin Properties
End Function
77 End Module
78
79
```

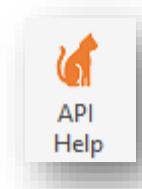
The code editor shows a snippet of VBA-like code:

```
    'omDrawingTab As RibbonTab, ribbonPanel As RibbonPanel,
    'InButtonStack As Boolean) As ButtonDefinition
    Inverter.ToIPictureDisp(My.Resources.Cat_32)
    -pConverter.ToIPictureDisp(My.Resources.Cat_16)

    'Help"
    'the button
    'Help"

    'nDef(environment, CustomDrawingTab, ribbonPanel, useLargeIcon,
    'isInButtonStack, useProgressToolTip,
    'buttonLabel, toolTip_Simple, toolTip_Expanded,
    'standardIcon, largeIcon, toolTipImage)
```

Specify the icons



# Understanding the Add-in Template Structure



The screenshot shows the Microsoft Visual Studio interface. On the left, the 'Resources' pane is open, displaying various image files: Cat\_16, Cat\_32, Dog\_16, Dog\_32, API\_ToolTip, Elephant\_16, and Elephant\_32. A red arrow points from the 'Resources' tab in the left sidebar to the pane itself. Another red arrow points from the 'Images' button in the top toolbar to the pane. A large green curved arrow at the bottom points from the 'Resources' pane towards the 'Solution Explorer' on the right.

My\_New\_Addin\* API\_Help.vb

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search... Live Share

My\_New\_Addin

Application Compile Debug References Services Settings Signing My Extensions Code Analysis

Resources\*

Images Add Resource Remove Resource Access Modifier: Friend

Cat\_16 Cat\_32 Dog\_16

Dog\_32 API\_ToolTip Elephant\_16

Elephant\_32

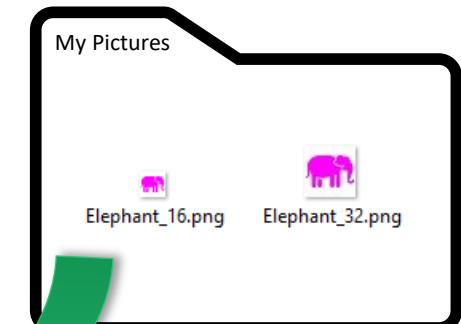
Solution Explorer

Search Solution Explorer (Ctrl+)

Solution 'My\_New\_Addin' (1 of 1 pr

- My\_New\_Addin
  - My Project
  - References
  - Command Tools
    - VB API\_Help.vb
    - VB Detect\_Dark\_Light\_Theme
    - VB Welcome.vb
  - Resources
    - API\_ToolTip.png
    - Cat\_16.png
    - Cat\_32.png
    - Dog\_16.png
    - Dog\_32.png
    - Elephant\_16.png
    - Elephant\_32.png
  - Utility Tools
    - VB AssemblyInfo.vb
    - Autodesk.My\_New\_Addin.manifest
    - My\_New\_Addin.X.manifest
    - Readme.txt
  - VB StandardAddInServer.vb

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



# Understanding the Add-in Template Structure

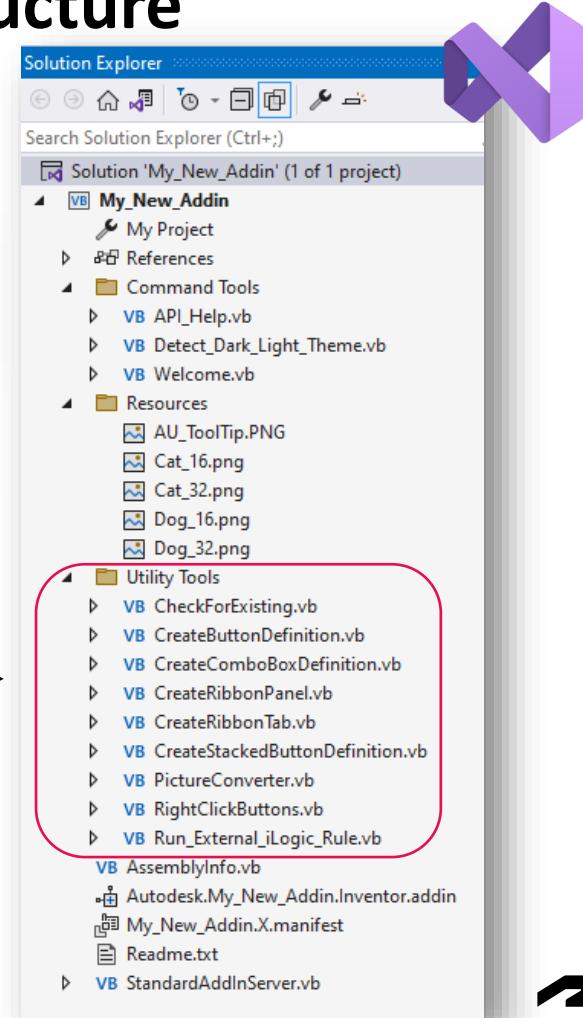
- Modules that are called over and over from the Command Tools and the StandardAddInServer

```
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")
```

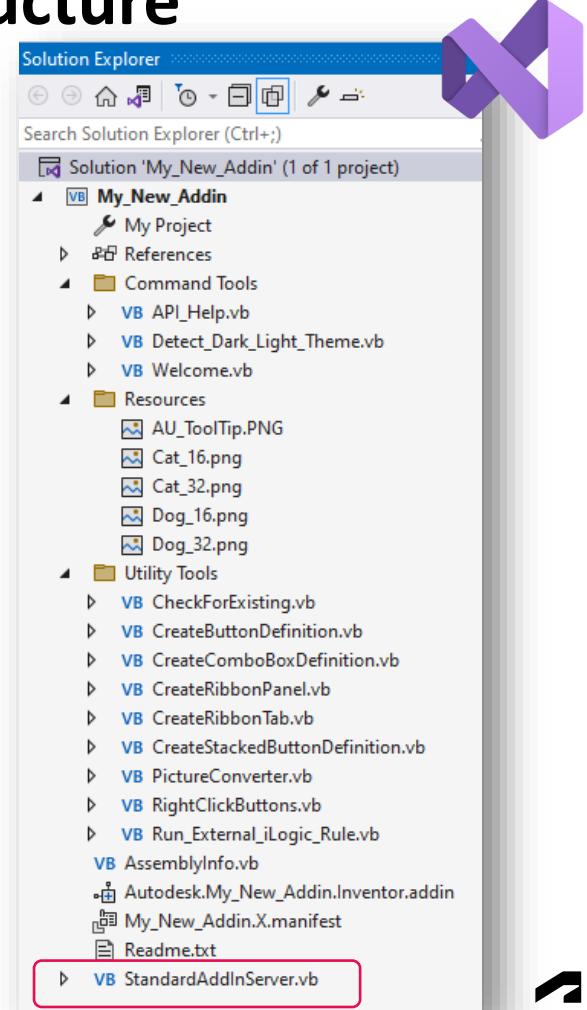
Utility Modules →



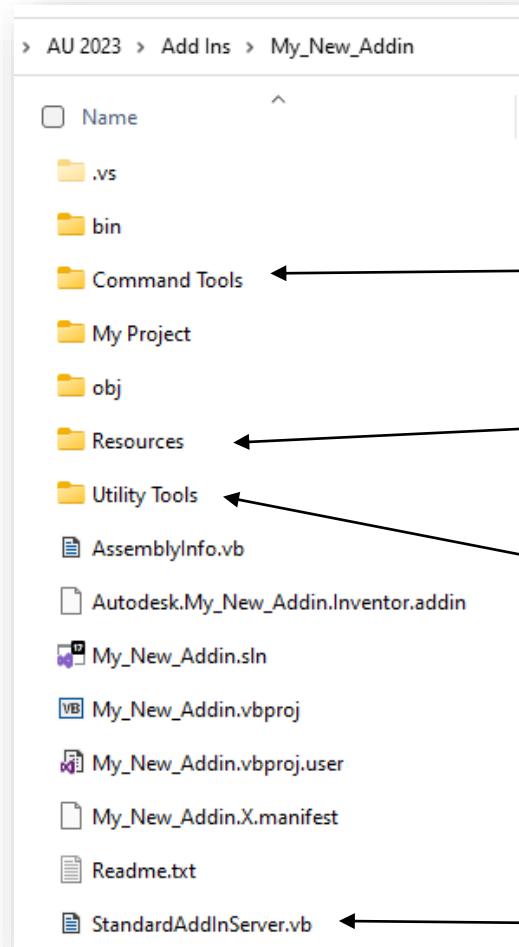
# Understanding the Add-in Template Structure

- 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

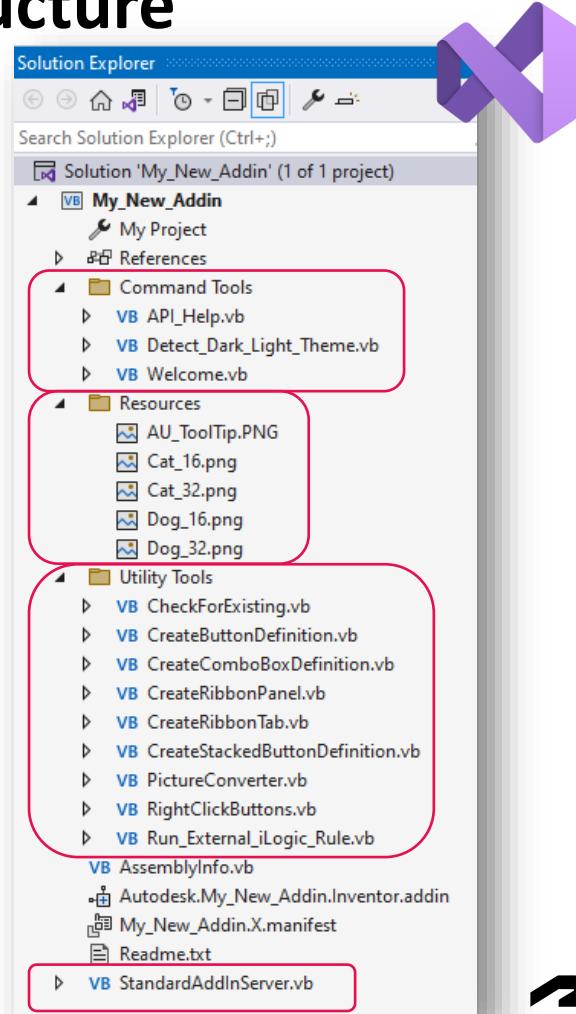


Command Tool Modules

Resources

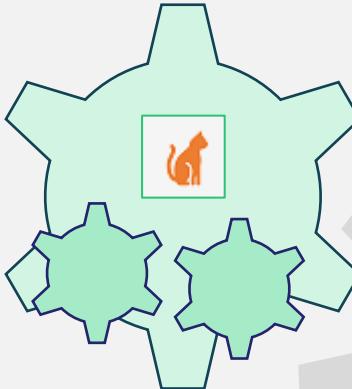
Utility Modules

Standard Add-in Server



# Understanding the Add-in Template Structure

Utilities



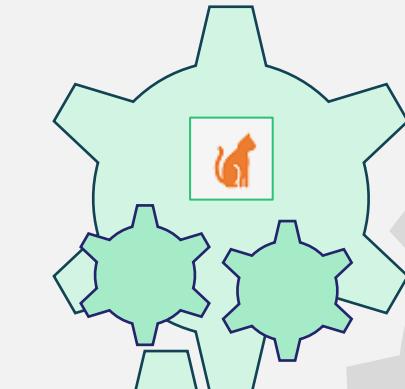
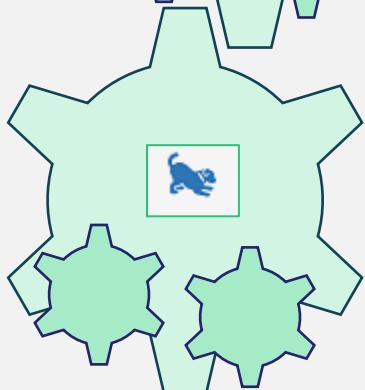
Standard Add-in  
Server



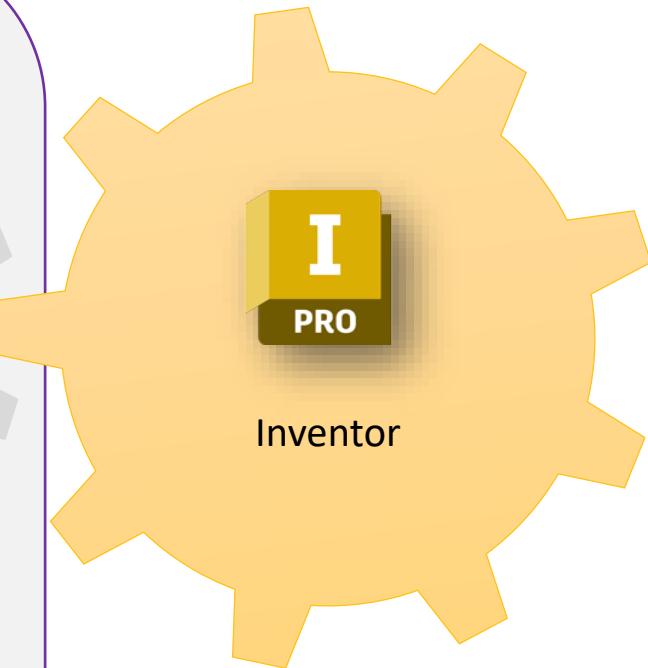
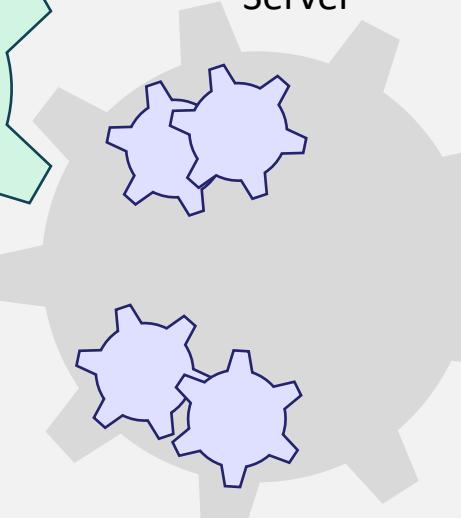
Inventor

# Understanding the Add-in Template Structure

Utilities



Standard Add-in  
Server

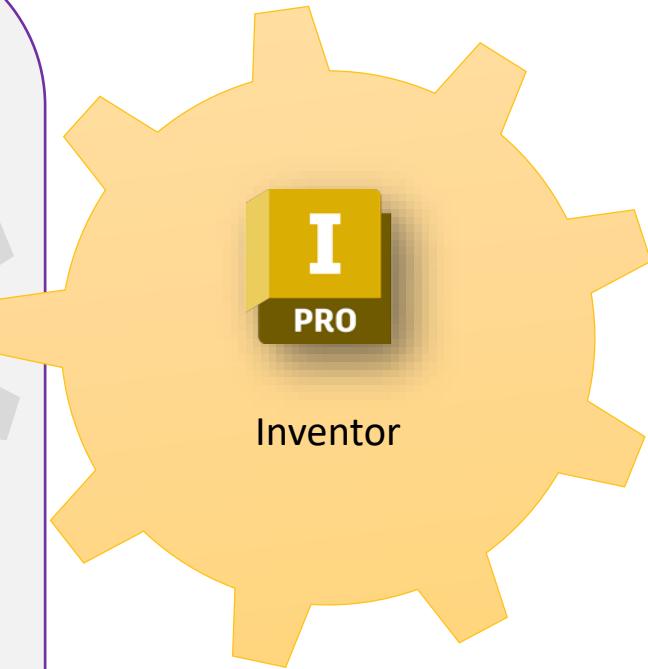
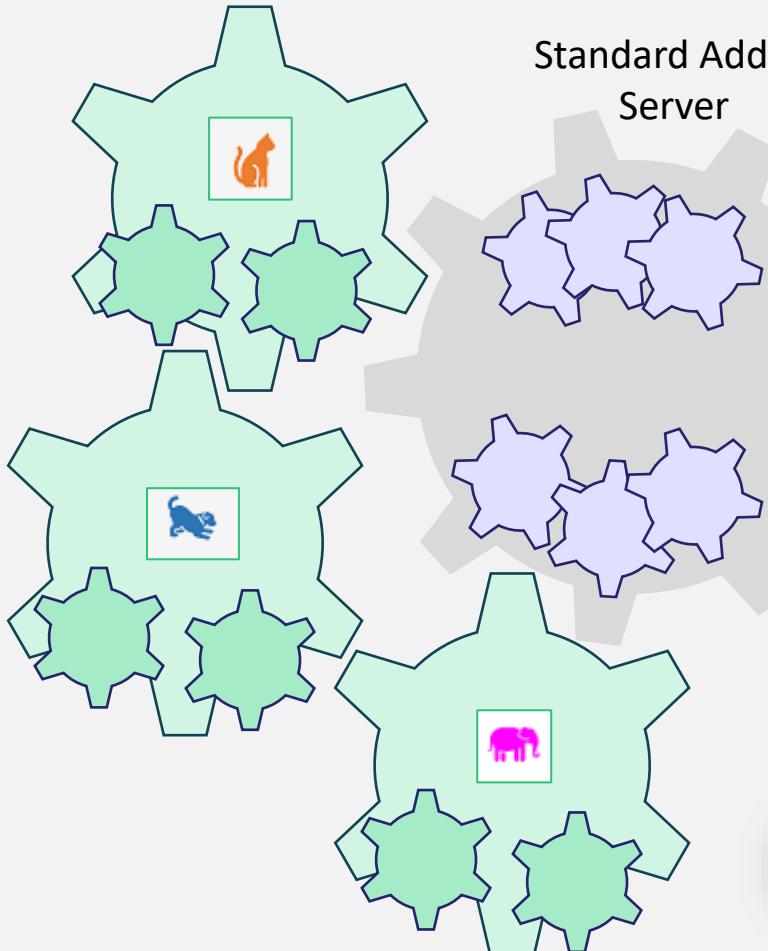


Inventor

# Understanding the Add-in Template Structure

Utilities

Standard Add-in  
Server



# Modifying the Add-in

Working in Visual Studio

# Setup Information



A screenshot of Microsoft Visual Studio showing the code for the StandardAddinServer file. The code defines several buttons and includes a region labeled "Setup Information". A red arrow points to the start of this region. A dashed box highlights the line of code that creates a list of environments: "Dim EnvironmentList As New List(Of String)({"Drawing", "Assembly", "Part"})".

```
Private WithEvents APS_Help_Button As ButtonHandlerFunction
Private WithEvents Detect_Dark_Light_Theme_Button As ButtonHandlerFunction
Private WithEvents Toggle_Sheet_Color As ButtonHandlerFunction
Private WithEvents Balloon_Notify_Button As ButtonHandlerFunction

Private Sub AddTabInterface()
    #Region "Setup Information"
        'create a object collection and Control Definition array to be used with button stack
        Dim buttonObjectCollection As ObjectCollection = g_inventorApplication.TransientObjects
        Dim ctrlDef(0 To 99) As ControlDefinition

        'create list of environments to cycle through
        Dim EnvironmentList As New List(Of String)( {"Drawing", "Assembly", "Part"} )

        'define custom ribbon tab prefix and suffix
        'this will be combined with the EnvironmentList to create ribbon tabs like "ACME Draw"
        Dim RibbonTabName_Prefix As String = "ACME"
        Dim RibbonTabName_Suffix As String = "Tools"

        'create list of environments to cycle through
        Dim EnvironmentList As New List(Of String)( {"Drawing", "Assembly", "Part"} )
    End Sub
```

- We double click the **StandardAddinServer** file in the Solution Explorer 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



The screenshot shows the Microsoft Visual Studio IDE with the StandardAddInServer.vb file open. The code is written in VB.NET and defines a ribbon tab setup. A callout box highlights the modification of the 'RibbonTabName\_Prefix' variable from 'ACME' to 'Vandelay'.

```
StandardAddInServer.vb*  StandardAddInServer  AddToUserInterface

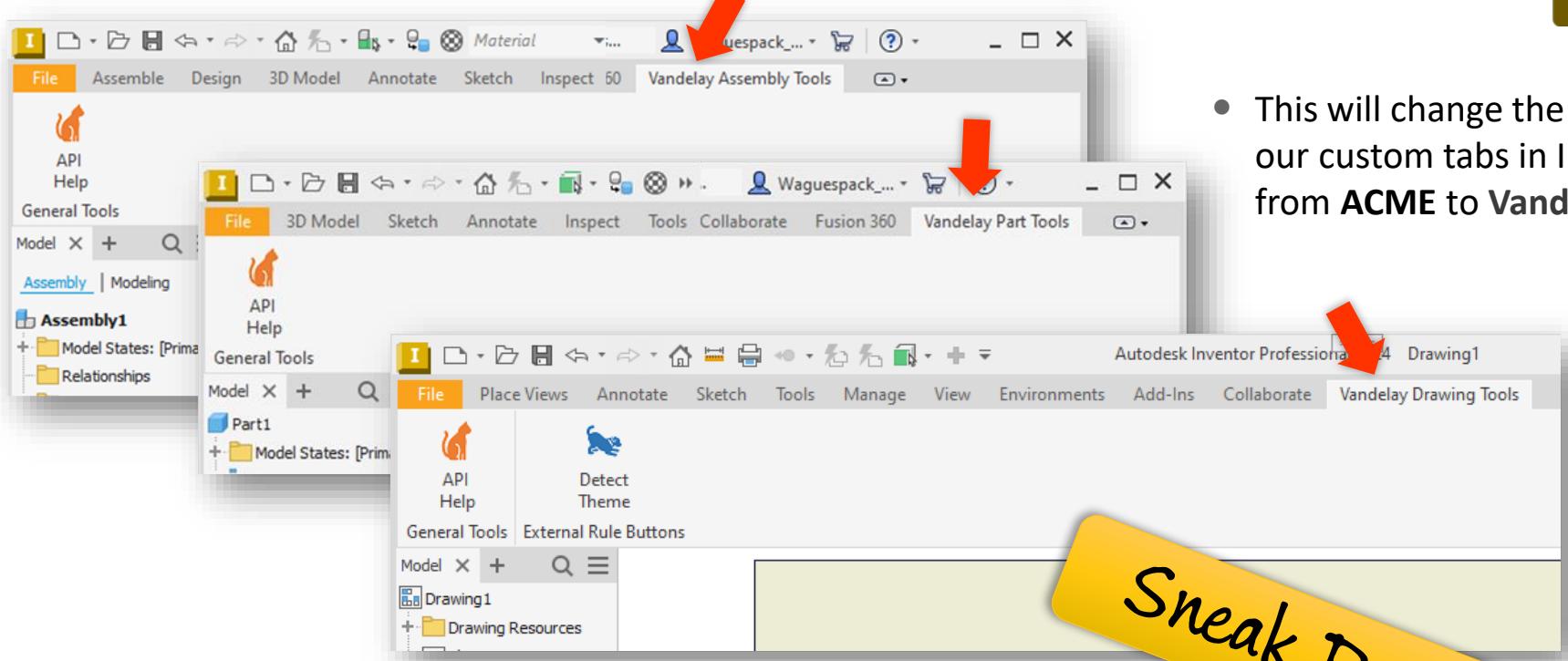
34     'Define the command buttons
35     Private WithEvents API_Help_Button As ButtonDefinition
36     Private WithEvents Detect_Dark_Light_Theme_Button As ButtonDefinition
37     Private WithEvents Toggle_Sheet_Color As ButtonDefinition
38     Private WithEvents Balloon_Notify_Button As ButtonDefinition
39
40     2 references
41     Private Sub AddToUserInterface()
42         #Region "Setup Information"
43             'create a object collection and Control Definition array to be used with button stack
44             Dim buttonObjectCollection As ObjectCollection = g_inventorApplication.TransientObject
45             Dim ctrlDef(0 To 99) As ControlDefinition
46
47             'create list of environments to cycle through
48             Dim EnvironmentList As New List(Of String)( {"Drawing", "Assembly", "Part"})
49
50             'define custom ribbon tab prefix and suffix
51             'this will be combined with the EnvironmentList to create ribbon tabs like "ACME Draw"
52             Dim RibbonTabName_Prefix As String = "ACME"
53             Dim RibbonTabName_Suffix As String = "Tools"
54
55             'create list of panels to create on the custom
56             Dim CustomPanelList As New List(Of String)( {"Ge
57         #End Region
58
59         Create Custom Tabs and Panels for each environment in the
126
127         Create Buttons
139
140             End Sub
142
143
144             Button 'on click' Events
155
100 %  No issues found  Add to Source Control  Select Repository  Ready
```

The code defines a ribbon tab setup with the following structure:

- It creates a `buttonObjectCollection` and `ctrlDef` arrays.
- It defines a `EnvironmentList` containing "Drawing", "Assembly", and "Part".
- It sets up custom ribbon tabs with a prefix of "ACME" and a suffix of "Tools".
- It creates a `CustomPanelList`.
- It ends the region and creates custom tabs and buttons for each environment in the `EnvironmentList`.
- It handles button click events.

- 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

The screenshot shows the Autodesk Inventor development environment. On the left, the Solution Explorer window displays the project structure for 'My\_New\_Addin'. The 'StandardAddInServer.vb' file is selected. On the right, the ribbon bar is visible with two tabs: 'General Tools' and 'External Rule Buttons'. Under 'General Tools', there are two buttons: 'API Help' (represented by a fox icon) and 'Detect Theme' (represented by a cat icon). A callout box points from the 'Create Buttons' section of the list to the 'External Rule Buttons' tab in the ribbon, indicating that the buttons shown in the ribbon were created in the 'StandardAddInServer.vb' file.

Solution Explorer

Solution 'My\_New\_Addin' (1 of 1 project)

- My\_New\_Addin
  - My Project
  - References
  - Command Tools
    - API\_Help.vb
    - Detect\_Dark\_Light\_Theme.vb
    - Welcome.vb
  - Resources
  - Utility Tools
  - AssemblyInfo.vb
  - Autodesk.My\_New\_Addin.Inventor.ad
  - My\_New\_Addin.X.manifest
  - Readme.txt
- StandardAddInServer.vb

Solution Explorer

Solution 'My\_New\_Addin' (1 of 1 project)

- My\_New\_Addin
  - My Project
  - References
  - Command Tools
    - API\_Help.vb
    - Detect\_Dark\_Light\_Theme.vb
    - Welcome.vb
  - Resources
  - Utility Tools
  - AssemblyInfo.vb
  - Autodesk.My\_New\_Addin.Inventor.ad
  - My\_New\_Addin.X.manifest
  - Readme.txt
- StandardAddInServer.vb

General Tools External Rule Buttons

API Help Detect Theme

I PRO



# Modifying the Add-in

The screenshot shows the Microsoft Visual Studio IDE interface. On the left, the code editor displays the `StandardAddinServer.vb` file under the `My_New_Addin` namespace. The code defines a class `StandardAddInServer` that implements `Inventor.ApplicationAddInServer`. It also declares several event handlers for application events like `WithEvents InventorApplicationEvents As ApplicationEvents`. In the middle section, there are comments for defining command buttons. On the right, the Solution Explorer pane shows the project structure for `'My_New_Addin'`, including files like `My_New_Addin.csproj`, `VB API_Help.vb`, `VB Detect_Dark_Light_Theme_Button.cs`, `VB Welcome.vb`, `Resources`, `Utility Tools`, and `Autodesk.My_New_Addin.manifest`.

```
Imports ...  
'Define global variables  
Public Module Globals ...  
  
Namespace My_New_Addin  
    <ProgIdAttribute("My_New_Addin.StandardAddInServer"),  
     GuidAttribute(g_simpleg_addInClientID)>  
    Public Class StandardAddInServer  
        Implements Inventor.ApplicationAddInServer  
  
        'Application events  
        Private WithEvents InventorApplicationEvents As ApplicationEvents  
        Private WithEvents UIEvents As UserInterfaceEvents  
        Private WithEvents UserInputEvents As UserInputEvents  
        Private WithEvents InvTransactionEvents As TransactionEvents  
  
        'Define the command buttons  
        Private WithEvents API_Help_Button As ButtonDefinition  
        Private WithEvents Detect_Dark_Light_Theme_Button As ButtonDefinition  
  
        2 references  
        Private Sub AddToUserInterface()  
  
        #Setup Information  
  
        #Create Custom Tabs and Panels for each environment in the list  
  
        #Create Single Buttons  
  
        End Sub
```

- 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



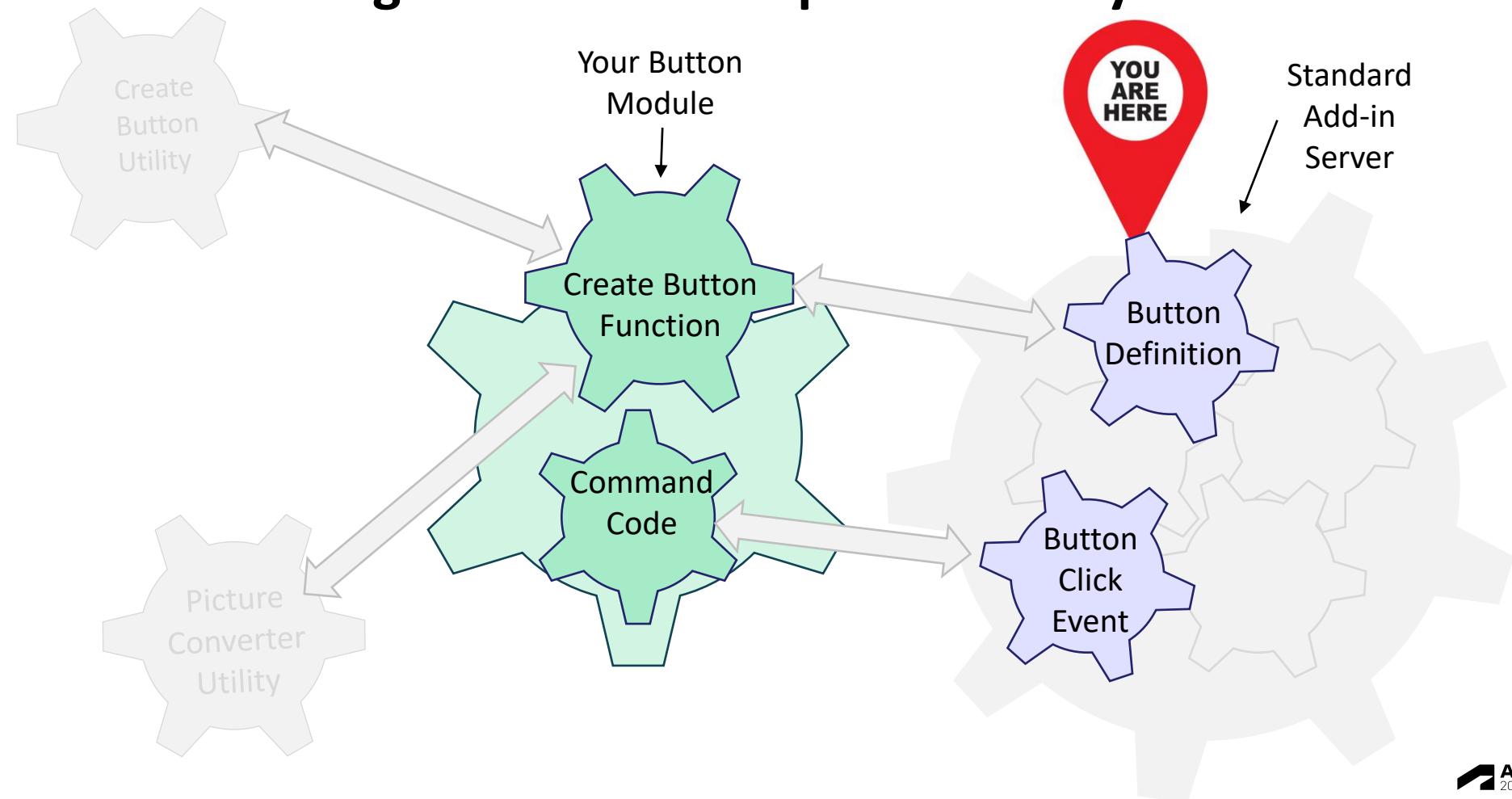
The screenshot shows the Microsoft Visual Studio interface. The code editor window displays the file `StandardAddInServer.vb`. The code defines a class `StandardAddInServer` that implements `Inventor.ApplicationAddInServer`. It includes event handlers for application events like `InventorApplicationEvents`, `UserInterfaceEvents`, `UserInputEvents`, and `InvTransactionEvents`. A new button definition is being added:

```
Private WithEvents Toggle_Sheet_Color_Button As ButtonDefinition
```

The Solution Explorer window shows the project structure for "My\_New\_Addin". The `StandardAddInServer.vb` file is selected. Other files in the project include `VB API_Help.vb`, `VB Detect_Dark_Light_Theme.vb`, `VB Welcome.vb`, `Resources`, `Utility Tools`, `Autodesk.My_New_Addin.manifest`, and `Readme.txt`.

- 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



```
VB My_New_Addin
 35  Private WithEvents API_Help_Button As ButtonDefinition
 36  Private WithEvents Detect_Dark_Light_Theme_Button As ButtonDefinition
 37  Private WithEvents Toggle_Sheet_Color_Button As ButtonDefinition
 38
 39
 40  Private Sub AddToUserInterface()
 41
 42  #Setup Information
 43
 44  #Create Custom Tabs and Panels for each environment in the list
 45
 46  #Region "Create Buttons"
 47
 48    'add this button to General Tools tab of 3 different environments
 49    API_Help_Button = API_Help.CreateButton("Drawing", CustomDrawingTab, Drawing_GeneralToolsPanel, True, False)
 50    API_Help_Button = API_Help.CreateButton("Assembly", CustomAssemblyTab, Assembly_GeneralToolsPanel, True, False)
 51    API_Help_Button = API_Help.CreateButton("Part", CustomPartTab, Part_GeneralToolsPanel, True, False)
 52
 53    'add button to just one tab
 54    Detect_Dark_Light_Theme_Button =
 55      Detect_Dark_Light_Theme.CreateButton("Drawing", CustomDrawingTab, Drawing_ExternalRuleButtonsPanel, True, False)
 56
 57  #End Region
 58
 59
 60  End Sub
 61
 62
 63
 64
 65  #Button 'on click' Events
 66
 67  #Application Events
 68
 69  #ApplicationAddInServer Members
 70
 71
 72
 73
 74
 75
 76
 77
 78
 79
 80
 81
 82
 83
 84
 85
 86
 87
 88
 89
 90
 91
 92
 93
 94
 95
 96
 97
 98
 99
 100
```

The image shows a screenshot of the Microsoft Visual Studio IDE. A red arrow points to the line of code '#Region "Create Buttons"' in the StandardAddInServer.vb file. The code editor displays several regions: 'Create Buttons', 'Button 'on click' Events', 'Application Events', and 'ApplicationAddInServer Members'. The 'Create Buttons' region contains code for creating buttons in three different environments: Drawing, Assembly, and Part. The 'Button 'on click' Events' region contains code for the Detect\_Dark\_Light\_Theme\_Button. The 'Application Events' and 'ApplicationAddInServer Members' regions are currently collapsed. The status bar at the bottom indicates 'Ready'.

# Modifying the Add-in



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

```
StandardAddInServer.vb*  StandardAddInServer
My_New_Addin
35  Define the 'Command Tools'
36  Private WithEvents API_Help_Button As ButtonDefinition
37  Private WithEvents Detect_Dark_Light_Theme_Button As ButtonDefinition
38  Private WithEvents Toggle_Sheet_Color_Button As ButtonDefinition
39
40  2 references
41  Private Sub AddToUserInterface()
42
43  #Setup Information
44
45  #Create Custom Tabs and Panels for each environment in the list
46
47  #Region "Create Buttons"
48
49  'add this button to General Tools tab of 3 different environments
50  API_Help_Button = API_Help.CreateButton("Drawing", CustomDrawingTab, Drawing_GeneralToolsPanel)
51  API_Help_Button = API_Help.CreateButton("Assembly", CustomAssemblyTab, Assembly_GeneralToolsPanel)
52  API_Help_Button = API_Help.CreateButton("Part", CustomPartTab, Part_GeneralToolsPanel)
53
54  #End Region
55
56  End Sub
57
58
59  #Button 'on click' Events
60
61  #Application Events
62
63  #ApplicationAddInServer Members
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
```

No issues found | Add to Source Control | Select Repository | Ready

# 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

The screenshot shows the Microsoft Visual Studio interface with the following details:

- Solution Explorer:** Displays the solution 'My\_New\_Addin' with one project 'My\_New\_Addin'. Inside the project, there are several files: 'My Project', 'References', 'Command Tools' (containing 'VB API\_Help.vb'), 'VB Detect\_Dark\_Light\_Theme.vb', 'VB Welcome.vb', 'Resources', 'Utilities', 'AssemblyInfo.vb', 'Autodesk.My\_New\_Addin.Inventor.addin', 'My\_New\_Addin.X.manifest', 'Readme.txt', and 'VB StandardAddInServer.vb'.
- Code Editor:** Shows the file 'StandardAddInServer.vb'. The code is written in VB.NET and includes sections for 'Setup Information', 'Create Custom Tabs and Panels for', and 'Create Buttons'. It defines several private members and methods, including 'AddToUserInterface()' and 'Detect\_Dark\_Light\_Theme()'.
- Context Menu:** A context menu is open over the 'VB API\_Help.vb' file in the Solution Explorer. The menu items include 'Open', 'Open With...', 'Code Cleanup', 'Run Tests', 'Debug Tests', 'View Code' (F7), 'Scope to This', 'New Solution Explorer View', 'Exclude From Project', 'Cut' (Ctrl+X), **Copy** (Ctrl+C) (highlighted with a red arrow), 'Delete' (Del), 'Rename' (F2), 'Copy Full Path', and 'Properties' (Alt+Enter).

# Modifying the Add-in



- Right click on the Command Tools folder and choose Paste

The screenshot shows the Microsoft Visual Studio interface. On the left is the StandardAddInServer.vb code editor, displaying VB.NET code for an Inventor add-in. In the center is the Solution Explorer window, which lists the project 'My\_New\_Addin' with its files: My\_New\_Addin.vb, References, Command Tools, and API.Help.vb. A context menu is open over the 'Command Tools' folder. The menu items include: Add, Run Tests, Debug Tests, Scope to This, New Solution Explorer View, Exclude From Project, Cut (Ctrl+X), Copy (Ctrl+C), Paste (highlighted with a red arrow), Paste As Link, Delete (Del), Rename (F2), Copy Full Path, Open Folder in File Explorer, Open in Terminal, Properties, and Alt+Enter. To the right of the Solution Explorer is another instance of the Solution Explorer showing the contents of the 'Command Tools' folder, which includes: Detect\_Dark\_Light\_Theme.vb, Welcome.vb, Resources, Utility Tools, AssemblyInfo.vb, Autodesk.My\_New\_Addin.Inventor.addin, My\_New\_Addin.X.manifest, Readme.txt, and StandardAddInServer.vb. A green arrow points from the 'Command Tools' folder in the main Solution Explorer to the 'Command Tools' folder in the secondary Solution Explorer.

# Modifying the Add-in

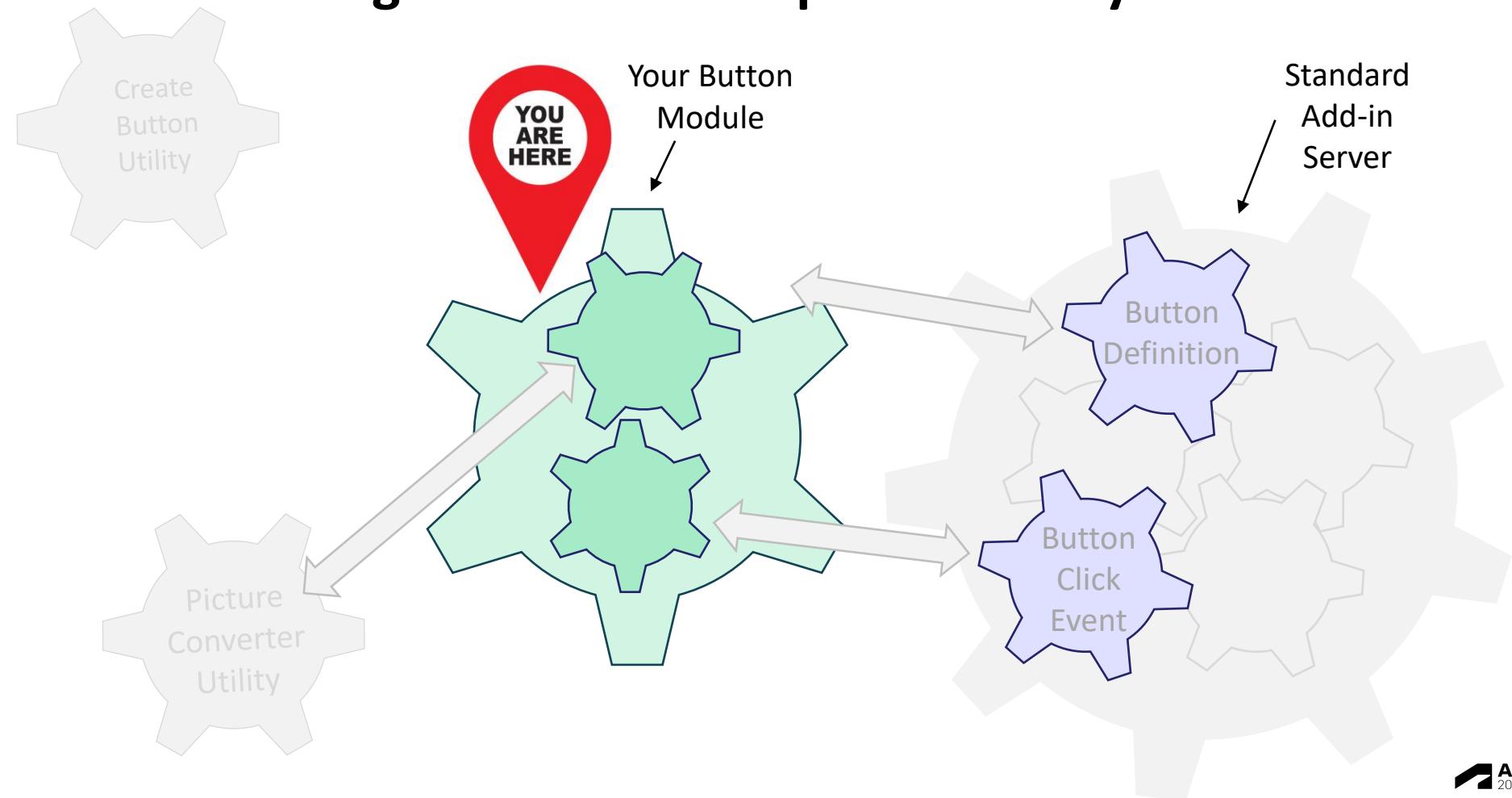


- Rename the copy

The screenshot shows the Microsoft Visual Studio interface with the following details:

- Solution Explorer:** Shows a single project named "My\_New\_Addin". Inside the project, there is a folder named "Command Tools" containing several files: "VB API\_Help - Copy.vb", "VB API\_Help.vb", "VB Detect\_Dark\_Light\_Theme.vb", and "VB Welcome.vb". There are also folders for "Resources" and "Utility Tools".
- Code Editor:** The main window displays the code for "StandardAddInServer.vb". The code includes sections for defining command buttons and adding them to the user interface. A specific section is highlighted with a yellow box and a red arrow pointing to it, which corresponds to the "VB API\_Help - Copy.vb" file in the Solution Explorer.
- Toolbars and Menus:** Standard Visual Studio menus like File, Edit, View, Project, Build, etc., are visible at the top. The toolbar below the menu bar includes icons for opening, saving, and running the project.
- Status Bar:** The bottom status bar shows "100 %", "No issues found", "Ln: 137 Ch: 1 SPC CRLF", and navigation icons.

# Understanding the Add-in Template Visually



# Modifying the Add-in



The screenshot shows the Microsoft Visual Studio IDE interface. The Solution Explorer on the right lists the project 'My\_New\_Addin' with its files: My\_New\_Addin.vb, References, Command Tools, Utility Tools, Resources, and StandardAddInServer.vb. A red arrow points from the Solution Explorer to the module name 'API\_Help' in the code editor. The code editor displays the file 'Toggle\_Sheet\_Color.vb' with the following content:

```
Imports Inventor

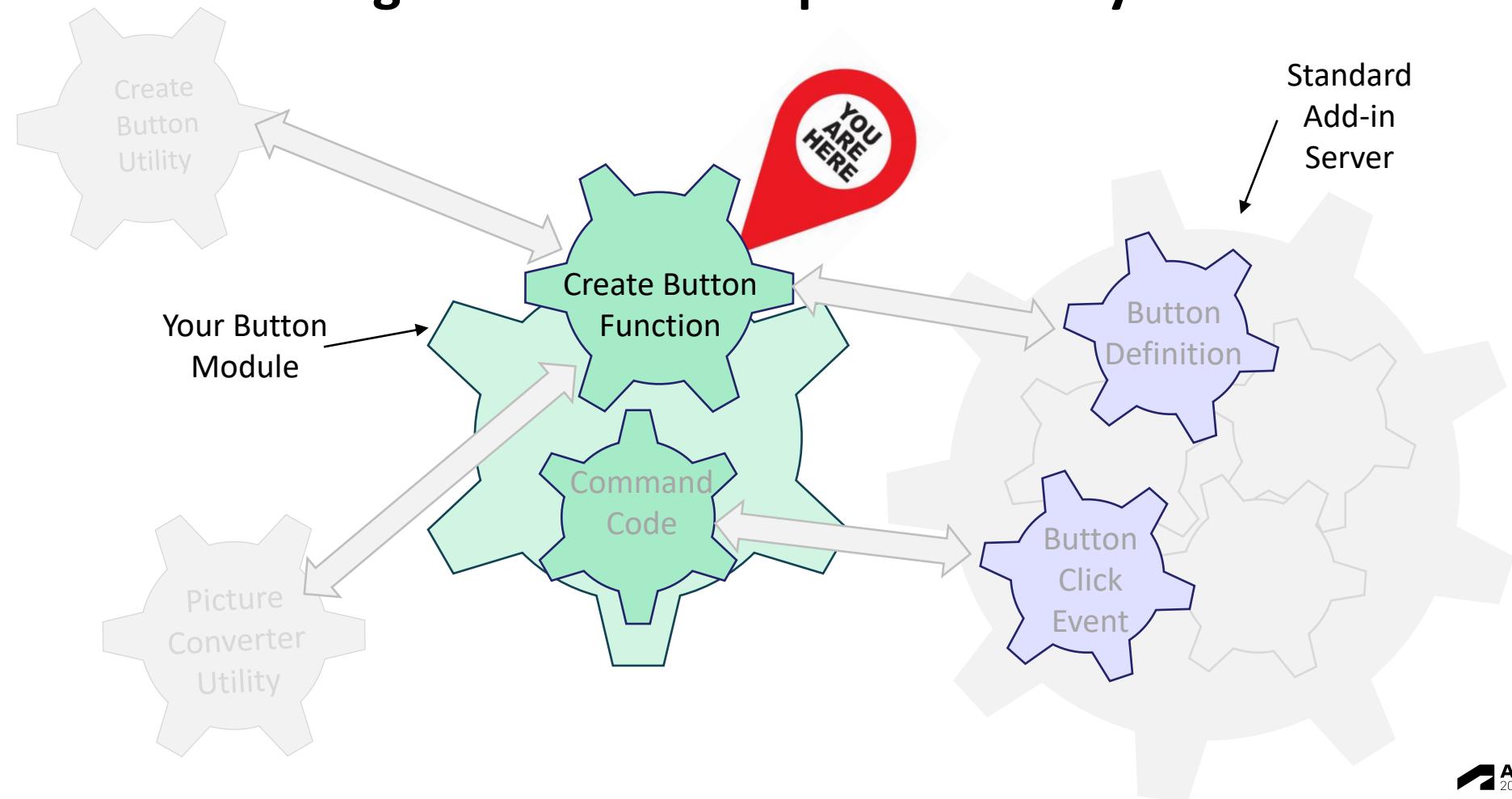
Module API_Help
    'Creates a button
    'Define the
    'Creates a button Button
    'This Function
    Function Create
        'get the information
        Dim largeIcon As IPictureDisp = PictureDispConverter.ToIPictureDisp(My.Resources)
        Dim standardIcon As IPictureDisp = PictureDispConverter.ToIPictureDisp(My.Resources)
        Dim toolTipImage As IPictureDisp = Nothing

        'this is the text the user sees on the button
        Dim buttonLabel As String = "API " & vbCrLf & "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
    End Function
End Module
```

- 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



A screenshot of the Microsoft Visual Studio IDE. The ribbon bar at the top has 'Project' highlighted with a red arrow pointing to it. Below the ribbon, the 'Solution Explorer' window shows a project named 'My\_New\_Addin' containing several files. The 'VB Toggle\_Sheet\_Color.vb' file is selected. In the code editor, there is a red dashed box highlighting a specific line of code:

```
    Interator.ToIPictureDisp(My.Resources.Cat_32)
    Interator.ToIPictureDisp(My.Resources.Cat_16)
```

The code editor also displays some descriptive comments above this line:

```
    'je As IPictureDisp = Nothing
    'ext the user sees on the button
    'L As String = "API " & vbCrLf & "Help"
    'lays when the user hovers over the button
    'ample As String = "Opens the API Help"
    'anded As String = Nothing
```

At the bottom of the screen, there is a status bar with the text 'Ready' and a progress bar.

- 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

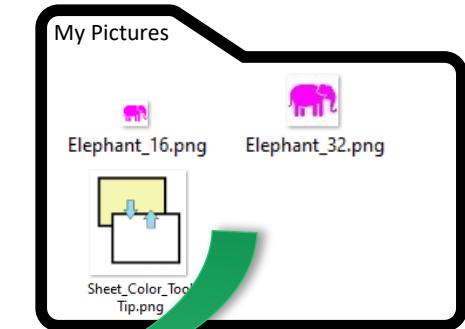


The screenshot shows the Microsoft Visual Studio interface with the 'My\_New\_Addin' project open. The 'Resources' tab is selected in the left sidebar. In the center, the 'Images' pane displays several icons: 'API\_ToolTip' (a help icon), 'Cat\_16' (orange cat), 'Cat\_32' (orange cat), 'Dog\_16' (blue dog), 'Dog\_32' (blue dog), 'Elephant\_16' (pink elephant), 'Elephant\_32' (pink elephant), and 'Sheet\_Color\_ToolTip' (a sheet of paper with a color palette icon). A red box highlights the 'API\_ToolTip' icon, and a red arrow points from it to the 'Resources' tab in the sidebar. A yellow sticky note in the top right corner contains the following text:

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

The Solution Explorer pane on the right lists the project files: 'My\_New\_Addin' (1 of 1 projects), 'My Project', 'References', 'Command Tools', 'VB API\_Help.vb', 'VB Detect\_Dark\_Light\_Theme.vb', 'VB Welcome.vb', 'Resources', and 'Utility Tools'. Under 'Resources', there are files: 'API\_ToolTip.png', 'Cat\_16.png', 'Cat\_32.png', 'Dog\_16.png', 'Dog\_32.png', 'Elephant\_16.png', 'Elephant\_32.png', and 'Utility Tools'. A red box highlights the 'Elephant\_16.png' and 'Elephant\_32.png' files. A large green curved arrow at the bottom right points from the 'Resources' pane towards a 'My Pictures' folder window.

- 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



A screenshot of the Microsoft Visual Studio IDE. The code editor window shows a portion of the file 'Toggle\_Sheet\_Color.vb' with the following code:

```
1
2
3     For
4         On Button_Definition
5
6             Disp(My.Resources.e)
7            ureDisp(My.Resources.e)
8
9             Disp(My.Resources.e)
10            ureDisp(My.Resources.e)
11
12             Disp(My.Resources.e)
13            ureDisp(My.Resources.e)
14
15             Disp(My.Resources.e)
16            ureDisp(My.Resources.e)
17
18             Disp(My.Resources.e)
19            ureDisp(My.Resources.e)
20
21             Disp(My.Resources.e)
22            ureDisp(My.Resources.e)
23
24             Disp(My.Resources.e)
25            ureDisp(My.Resources.e)
26
27             Disp(My.Resources.e)
28            ureDisp(My.Resources.e)
29
30             Disp(My.Resources.e)
31            ureDisp(My.Resources.e)
```

The 'Solution Explorer' window on the right lists the project structure:

- My\_New\_Addin (My Project)
- References
- Command Tools
  - API\_Help.vb
  - Detect\_Dark\_Light\_Theme.vb
  - Toggle\_Sheet\_Color.vb
  - Welcome.vb
- Resources
- Utility Tools
- AssemblyInfo.vb
- Autodesk.My\_New\_Addin.lnk
- My\_New\_Addin.X.manifest
- Readme.txt
- StandardAddinServer.vb

- 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



The screenshot shows the Microsoft Visual Studio IDE interface. The top menu bar includes File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help, and a search bar. The toolbar below has icons for file operations like Open, Save, and Print. The main code editor window displays VB.NET code for a Microsoft Word add-in. The code defines two buttons: one for toggling sheet colors and another for opening API help. The 'buttonLabel' variable is set to "Toggle " & vbCrLf & "Sheet Color" or "API " & vbCrLf & "Help". The 'toolTip\_Simple' variable is set to "Toggle the drawing sheet color" or "Opens the API Help". The 'toolTip\_Expanded' variable is set to Nothing. The code uses the 'CreateButtonDefintion' class to create button definitions. The Solution Explorer on the right shows the project structure with files like 'My\_New\_Addin.vb', 'StandardAddinServer.vb', and 'New\_Addin.manifest'. The status bar at the bottom shows the current line (Ln: 46), character (Ch: 12), column (Col: 20), and file type (SPC CRLF).

```
Imports Inventor

Module Toggle_Sheet_Color
    'this is the text the user sees on the button
    Dim buttonLabel As String = "Toggle " & vbCrLf & "Sheet Color"

    'text that displays when the user hovers over the button
    Dim toolTip_Simple As String = "Toggle the drawing sheet color"
    Dim toolTip_Expanded As String = Nothing

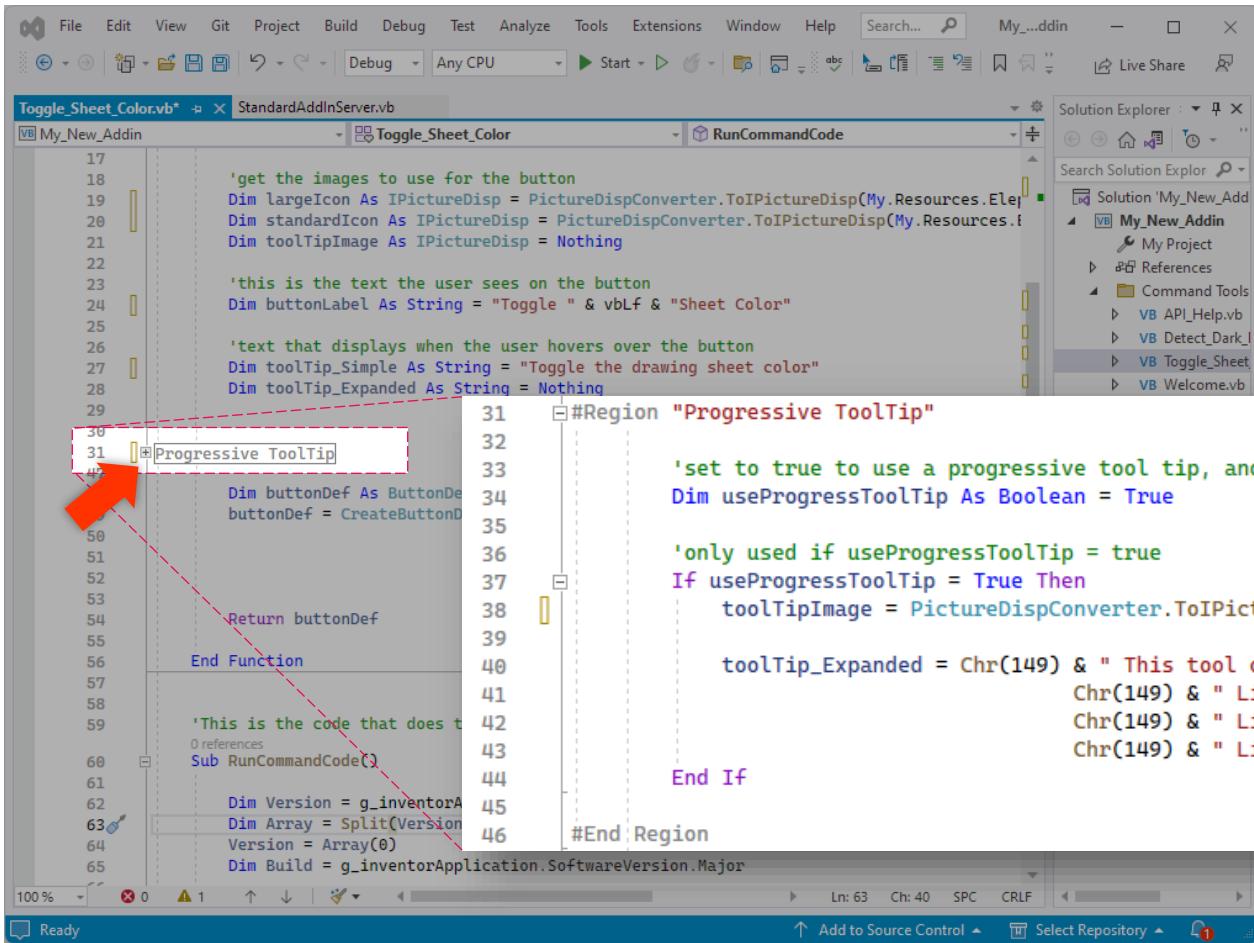
    'get the images to use for the button
    Dim largeIcon As IPictureDisp = PictureDispConverter.ToIPictureDisp(My.Resources.k_Light_Theme_Sheet_Color)
    Dim standardIcon As IPictureDisp = PictureDispConverter.ToIPictureDisp(My.Resources.k_Dark_Theme_Sheet_Color)
    Dim toolTipImage As IPictureDisp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

    'this is the text the user sees on the button
    Dim buttonLabel As String = "API " & vbCrLf & "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
End Module
```

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

# Modifying the Add-in



```
17
18
19     'get the images to use for the button
20     Dim largeIcon As IPictureDisp = PictureDispConverter.ToIPictureDisp(My.Resources.Ele...
21     Dim standardIcon As IPictureDisp = PictureDispConverter.ToIPictureDisp(My.Resources.I...
22     Dim tooltipImage As IPictureDisp = Nothing
23
24     'this is the text the user sees on the button
25     Dim buttonLabel As String = "Toggle " & vbCrLf & "Sheet Color"
26
27     'text that displays when the user hovers over the button
28     Dim toolTip_Simple As String = "Toggle the drawing sheet color"
29     Dim toolTip_Expanded As String = Nothing
30
31     #Region "Progressive ToolTip"
32
33         'set to true to use a progressive tool tip, and false to a simple tool tip
34         Dim useProgressToolTip As Boolean = True
35
36         'only used if useProgressToolTip = true
37         If useProgressToolTip = True Then
38             tooltipImage = PictureDispConverter.ToIPictureDisp(My.Resources.API_ToolTip)
39
40             toolTip_Expanded = Chr(149) & " This tool opens the API help *.chm file in a se...
41             Chr(149) & " Line2" & vbCrLf &
42             Chr(149) & " Line3" & vbCrLf &
43             Chr(149) & " Line4"
44
45         End If
46     #End Region
47
48     'This is the code that does t
49     0 references
50     Sub RunCommandCode()
51
52         Dim Version = g_inventorA...
53         Dim Array = Split(Version)
54         Version = Array(0)
55         Dim Build = g_inventorApplication.SoftwareVersion.Major
56
57
58
59
60
61
62
63
64
65
```

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

# Modifying the Add-in



The screenshot shows the Microsoft Visual Studio IDE with the following details:

- Title Bar:** File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help, Search (Ctrl+Q).
- Toolbars:** Standard, Debug, Start, Stop, Break, Task List, Solution Explorer, Properties, Task List, Status Bar.
- Code Editor:** The file `Toggle_Sheet_Color.vb` is open. The code is written in VB.NET. A yellow box highlights the tooltip text:

```
'only used if useProgressToolTip = true
If useProgressToolTip = True Then
    toolTipImage = PictureDispConverter.ToIPictureDisp(My.Resources.Sheet_Color_ToolTip)

    toolTip_Expanded = Chr(149) & " This tool toggles the sheet color" & vbCrLf &
                      Chr(149) & " Click once to change color" & vbCrLf &
                      Chr(149) & " Again to change it back"
End If
```
- Code Block:** A red dashed box encloses another section of code:

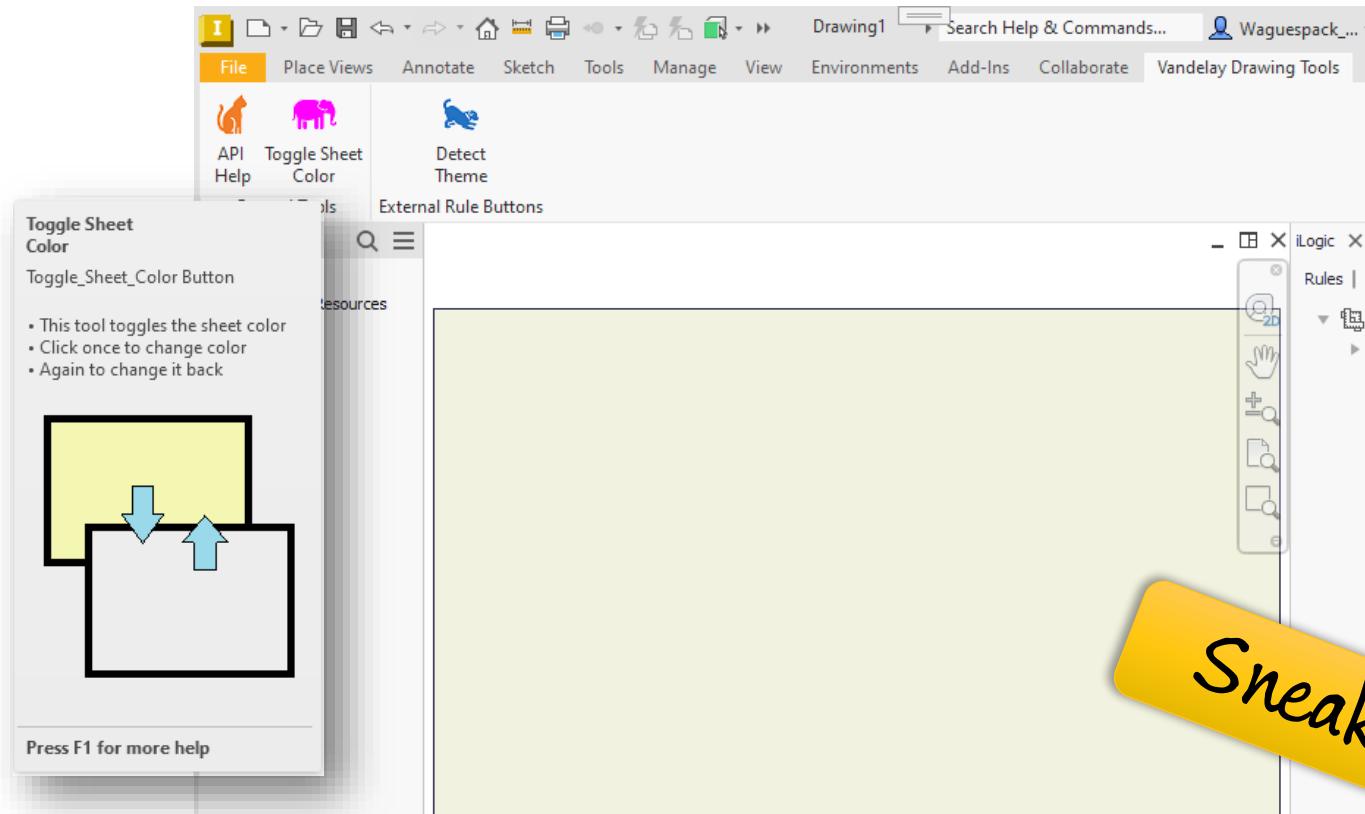
```
Dim useProgressToolTip As Boolean = True

'only used if useProgressToolTip = true
If useProgressToolTip = True Then
    toolTipImage = PictureDispConverter.ToIPictureDisp(My.Resources.API_ToolTip)

    toolTip_Expanded = Chr(149) & " This tool opens the API help *.chm file in a seperate window" & vbCrLf &
                      " Use the API help to find:" & vbCrLf &
                      Chr(149) & "      API Object Model Reference Information" & vbCrLf &
                      Chr(149) & "      Example Code"
End If
```
- Status Bar:** 100%, X 0, A 1, Up, Down, Left, Right, Ln: 24, Ch: 69, SPC, CRLF.
- Bottom Bar:** Ready, Add to Source Control, Select Repository, 1.

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

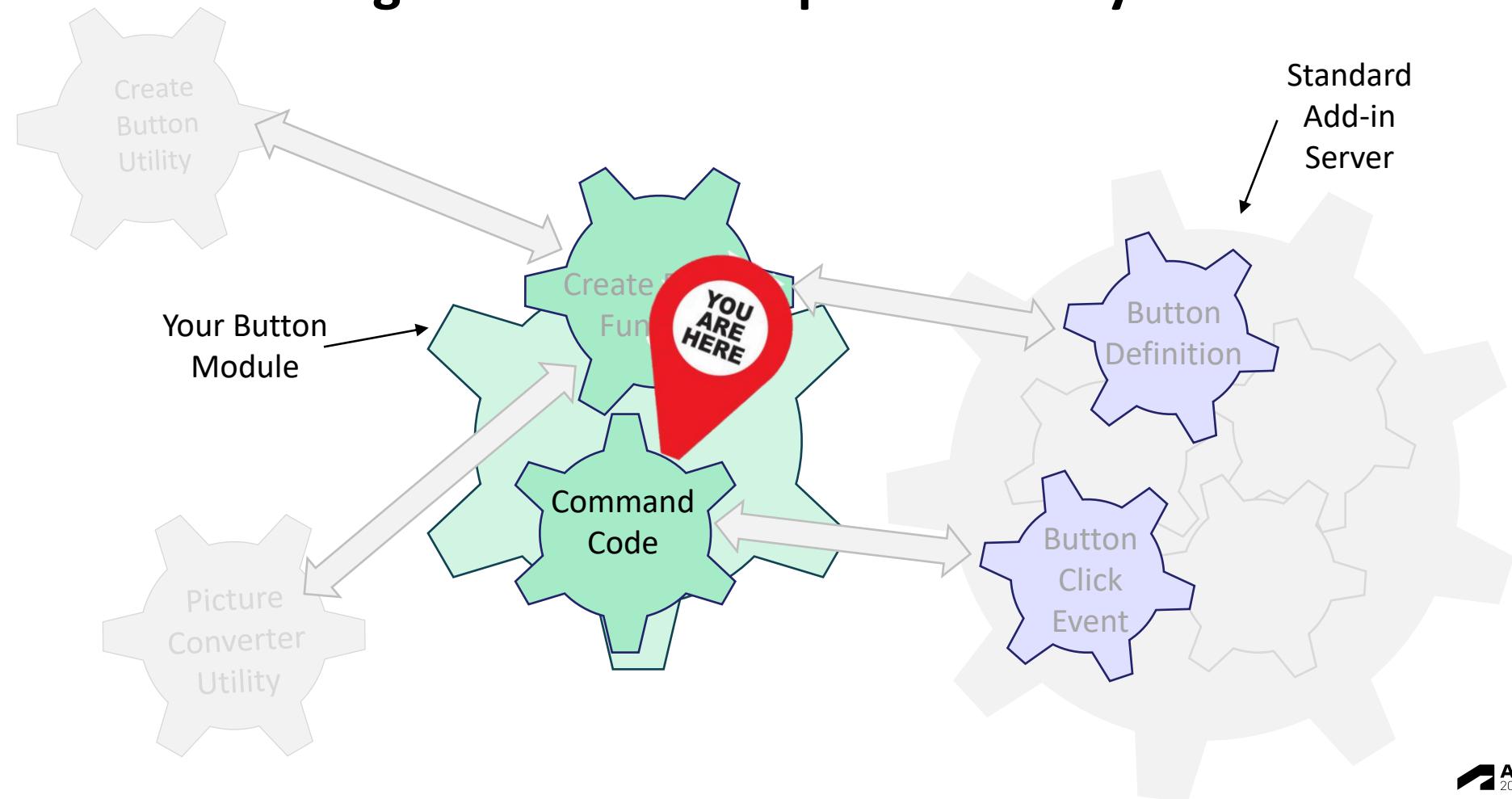
# Modifying the Add-in



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

Sneak Peak

# Understanding the Add-in Template Visually



# Modifying the Add-in



The screenshot shows the Microsoft Visual Studio IDE interface. The title bar says "My\_New\_Addin". The menu bar includes File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help, and a search bar. The toolbar has various icons for file operations. The main window shows the code editor with the file "Toggle\_Sheet\_Color.vb" open. The code defines a function "CreateButton" and a subroutine "RunCommandCode". The "RunCommandCode" subroutine contains logic to check if a specific help file exists and process it if found. A large red X is drawn over the entire content of the "RunCommandCode" subroutine. To the right is the Solution Explorer pane, which lists the solution "My\_New\_Addin" with one project "My\_New\_Addin" containing files like "API\_Help.vb", "Detect\_Dark\_Light\_Theme.vb", "Toggle\_Sheet\_Color.vb", "Welcome.vb", and various resource files. The bottom status bar shows "100% 0 1 Ln: 33 Ch: 77 SPC CRLF".

```
49
50
51
52
53     Return buttonDef
54
55 End Function
56
57
58 'This is the code that does the real work when your command is executed.
59 Sub RunCommandCode()
60
61     Dim Version = g_inventorApplication.SoftwareVersion.DisplayVersion
62     Dim Array = Split(Version, ".")
63     Version = Array(0)
64     Dim Build = g_inventorApplication.SoftwareVersion.Major
65
66     Dim Filename = "C:\Users\Public\Documents\Autodesk\Inventor " &
67     Version & "\Local Help\ADMAPI_" & Build & ".chm"
68
69     If System.IO.File.Exists(Filename) = True Then
70         Process.Start(Filename)
71     Else
72         MsgBox("File Does Not Exist" & vbCrLf & Filename)
73     End If
74
75 End Sub
76
77 End Module
78
```

- 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



The screenshot shows the Microsoft Visual Studio interface with a VBA project named "My\_New\_Addin". The code editor window displays the file "Toggle\_Sheet\_Color.vb" containing the following code:

```
49
50
51
52
53     Return buttonDef
54
55 End Function
56
57
58 'This is the code that does the real work when your command is executed.
59 Sub RunCommandCode()
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74     End Sub
75
76 End Module
77
78
```

The Solution Explorer window on the right lists the project structure:

- My\_New\_Addin (Project)
  - References
  - Command Tools
    - VB API\_Help.vb
    - VB Detect\_Dark\_Light\_Theme.vb
    - VB Toggle\_Sheet\_Color.vb
    - VB Welcome.vb
  - Resources
    - API\_ToolTip.png
    - Cat\_16.png
    - Cat\_32.png
    - Dog\_16.png
    - Dog\_32.png
    - Elephant\_16.png
    - Elephant\_32.png
    - Sheet\_Color\_ToolTip.png
  - Utility Tools
    - VB AssemblyInfo.vb
    - Autodesk.My\_New\_Addin.Inventor
    - My\_New\_Addin.X.manifest
    - Readme.txt
  - VB StandardAddInServer.vb

- 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

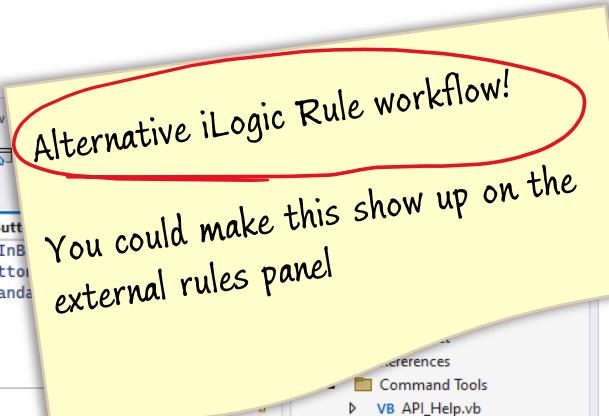


The screenshot shows the Microsoft Visual Studio interface. The code editor window displays a VB.NET module named 'Toggle\_Sheet\_Color.vb'. The code implements a command that toggles the sheet color of the active document between white and a dark light theme. The Solution Explorer window shows the project structure, including files like 'My\_New\_Addin.vb', 'VB API\_Help.vb', 'VB Detect\_Dark\_Light\_Theme.vb', 'VB Toggle\_Sheet\_Color.vb', 'VB Welcome.vb', and various resource files such as 'API\_ToolTip.png' and 'Sheet\_Color\_ToolTip.png'. The status bar at the bottom indicates the code is 100% complete.

```
56
57
58     'This is the code that does the real work when your command is executed.
59     0 references
60 Sub RunCommandCode()
61
62     Dim Doc As DrawingDocument = g_inventorApplication.ActiveDocument
63
64     Dim TObj As TransientObjects = g_inventorApplication.TransientObjects
65
66     'get the color of the sheet
67     Dim Color As Color
68     Color = Doc.SheetSettings.SheetColor
69
70     'get the RGB from the color
71     Dim RGB As String
72     RGB = Color.Red & ", " & Color.Green & ", " & Color.Blue
73
74     'toggle sheet color
75     'if white, set to inventor default color
76     If RGB = "255, 255, 255" Then
77         Color = TObj.CreateColor(237, 237, 214)
78         'if inventor default color, set to white
79     ElseIf RGB = "237, 237, 214" Then
80         Color = TObj.CreateColor(255, 255, 255)
81     Else
82         'if something else set to inventor default color
83         Color = TObj.CreateColor(237, 237, 214)
84     End If
85
86     Doc.SheetSettings.SheetColor = Color
87
88 End Sub
89
End Module
```

- 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



```
File Edit View Git Project Build Debug Test Analyze Tools Extensions Window
Toggle_Sheet_Color.vb* | Debug Any CPU Start
VB My_New_Addin Toggle_Sheet_Color CreateButton
49
50
51
52
53
54
55
56
57
58     Return buttonDef
End Function

' This is the code that does the real work when your command is executed.
0 references
Sub RunCommandCode()
    ' This is the code that does the real work when your command is executed.
    0 references
    Sub Run_ExternalRule()
        Run_External_iLogic_Rule.RunExternalRule("Toggle Sheet Color")
    End Sub
End Sub

End Module
```

The screenshot shows the Microsoft Visual Studio IDE interface. A yellow sticky note is overlaid on the code editor with the handwritten text: "Alternative iLogic Rule workflow! You could make this show up on the external rules panel". The code in the editor is written in VB.NET. It contains two subroutines: "RunCommandCode" and "Run\_ExternalRule". The "RunCommandCode" subroutine is crossed out with a large red X. The "Run\_ExternalRule" subroutine calls the "RunExternalRule" method of the "Run\_External\_iLogic\_Rule" class with the argument "Toggle Sheet Color". The project navigation pane on the right shows files like "VB API\_Help.vb", "VB Detect\_Dark\_Light\_Theme.vb", and "VB Toggle\_Sheet\_Color.vb". The status bar at the bottom indicates "Ln: 33 Ch: 77 SPC CRLF".

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

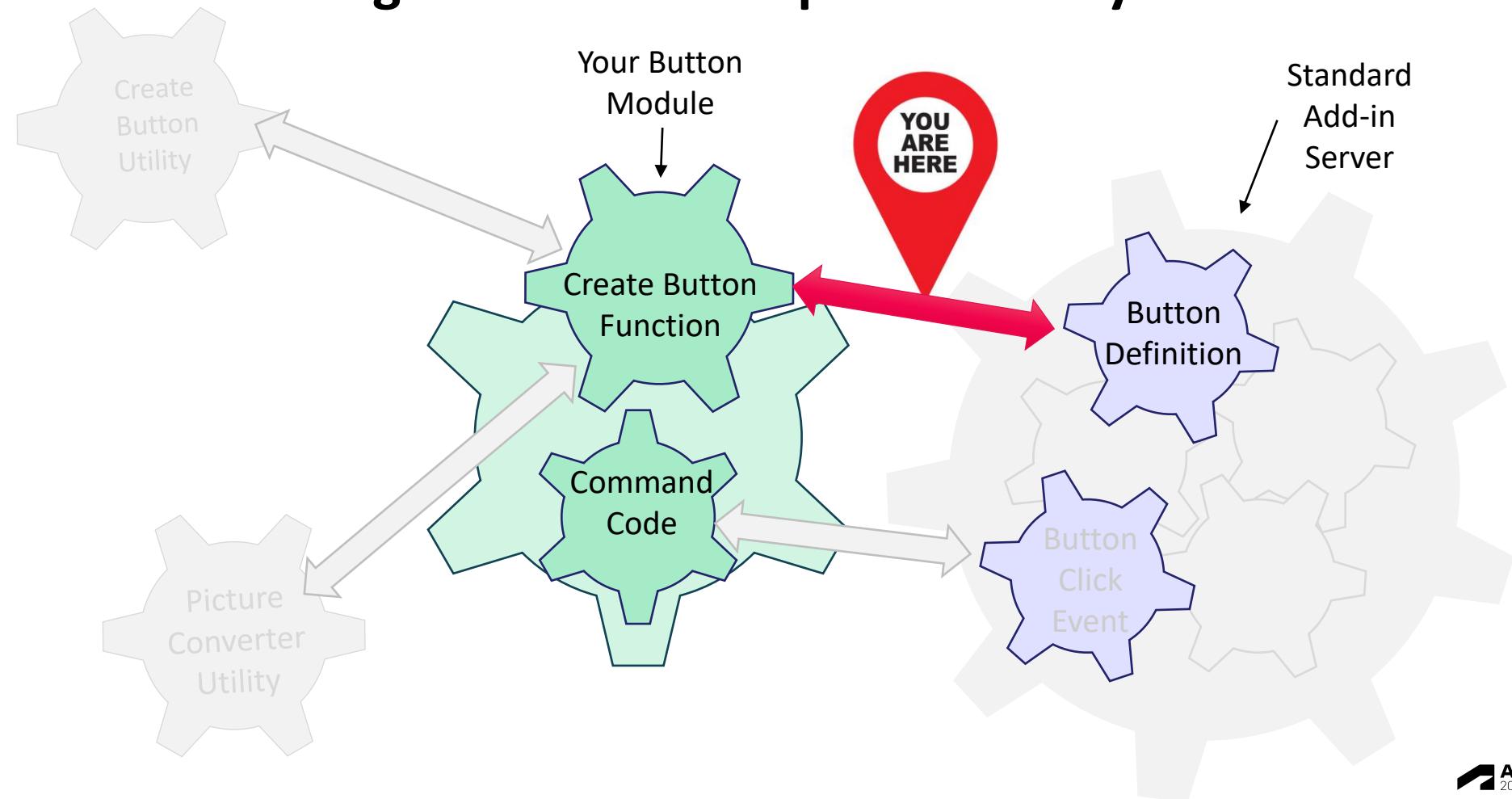


The screenshot shows the Visual Studio IDE with the following details:

- Title Bar:** File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help, Search (Ctrl+Q), My...addin.
- Toolbar:** Standard icons like New, Open, Save, Print, etc.
- Code Editor:** The file `Toggle_Sheet_Color.vb` is open. The code implements a command to toggle the sheet color in Inventor. It uses the `g_inventorApplication` object to get the active document and transient objects. It then checks the current sheet color and creates a new color with the opposite RGB values (e.g., white to black). Finally, it sets the sheet color to the newly created color.
- Solution Explorer:** Shows the project `My_New_Addin` with files `My_New_Addin.vb`, `My Project`, and `References`.
- Save Dialog:** A modal dialog box titled "Save changes to the following items?" is displayed, listing `Command Tools\Toggle_Sheet_Color.vb*`. It has three buttons: `Save`, `Don't Save`, and `Cancel`. A red arrow points to the `Save` button.
- Status Bar:** Shows 100%, 0 errors, and various status indicators like Ln: 55, Ch: 17, SPC, CRLF.
- Bottom Bar:** Ready, Add to Source Control, Select Repository, and a bell icon.

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

# Understanding the Add-in Template Visually



# Modifying the Add-in



The screenshot shows the Microsoft Visual Studio IDE interface with the following details:

- Menu Bar:** File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help.
- Search Bar:** Search (Ctrl+Q).
- Solution Explorer:** Shows the solution "My\_New\_Addin" with files like My\_New\_Addin.vb, VB API\_Help.vb, VB Detect\_Dark\_Light\_Theme.vb, VB Toggle\_Sheet\_Color.vb, VB Welcome.vb, Resources, Utility Tools, and VB AssemblyInfo.vb.
- Code Editor:** The file StandardAddInServer.vb\* is open, showing VBA-like code for a Microsoft Word add-in.

```
StandardAddInServer.vb*  Toggle_Sheet_Color.vb

VB My_New_Addin
    StandardAddInServer
        AddToUserInterface
            33
            34     'Define the command buttons
            35     Private WithEvents API_Help_Button As ButtonDefinition
            36     Private WithEvents Detect_Dark_Light_Theme_Button As ButtonDefinition
            37     Private WithEvents Toggle_Sheet_Color_Button As ButtonDefinition
            38
            39     Private Sub AddToUserInterface()
            40
            41     #Setup Information
            57
            58     #Create Custom Tabs and Panels for each environment in the list
            125
            126     #Region "Create Buttons"
            127
            128         'Add this button to CustomTools tab of 3 different environments
            129
            130         Toggle_Sheet_Color_Button =
            131             Toggle_Sheet_Color.CreateButton("Drawing", CustomDrawingTab,
            132                                         Drawing_GeneralToolsPanel, True, False)
            133
            134         Detect_Dark_Light_Theme_Button =
            135             Detect_Dark_Light_Theme.CreateButton("Drawing", CustomDrawingTab,
            136                                         Drawing_ExternalRuleButtonsPanel, True, False)
            137
            138         #End Region
            139
            140     End Sub
            141
            142
            143
            144     #Region "Button 'on click' Events"
            145
            146         'Add button click events to this region
            147
            148
            149
            150
```
- Status Bar:** Ln: 39 Ch: 41 SPC CRLF.
- Bottom Bar:** Ready, Add to Source Control, Select Repository.

- 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



```
StandardAddInServer.vb*  Toggle_Sheet_Color.vb
VB My_New_Addin  StandardAddInServer  AddToUserInterface
33
34
35     'Define the command buttons
36     Private WithEvents API_Help_Button As ButtonDefinition
37     Private WithEvents Detect_Dark_Light_Theme_Button As ButtonDefinition
38
39     Private Sub AddToUserInterface()
40
41     #Setup Information
42
43     #Create Custom Tabs and Panels for each environment in the list
44
45     #Region "Create Buttons"
46
47         'add this button to General Tools tab of 3 different environments
48         API_Help_Button = API_Help.CreateButton("Drawing", CustomDrawingTab, Drawing_GeneralToolsPanel)
49         API_Help_Button = API_Help.CreateButton("Assembly", CustomAssemblyTab, Assembly_GeneralToolsPanel)
50         API_Help_Button = API_Help.CreateButton("Part", CustomPartTab, Part_GeneralToolsPanel, True)
51
52         'add button to just one tab
53         Detect_Dark_Light_Theme_Button =
54             Detect_Dark_Light_Theme.CreateButton("Drawing", CustomDrawingTab,
55
56             Toggle_Sheet_Color_Button =
57                 Toggle_Sheet_Color.CreateButton("Drawing", CustomDrawingTab,
58                     Drawing_GeneralToolsPanel, True, False)
59
60     End Sub
61
62     #Region "Button 'on click' Events"
63
64     'Add button click events to this region
65
66 End Region
67
68 End Class
```

- 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



Alternative iLogic Rule workflow!

You could make this show up on the external rules panel

```
StandardAddInServer.vb  Toggle_Sheet_Color.vb
VB My_New_Addin  StandardAddInServer

33
34
35     'Define the command buttons
36     Private WithEvents API_Help_Button As ButtonDefinition
37     Private WithEvents Detect_Dark_Light_Theme_Button As ButtonDefinition
38
39     Private WithEvents Toggle_Sheet_Color_Button As ButtonDefinition
40
41     Private Sub AddToUserInterface()
42
43         'Setup Information
44
45         'Create Custom Tabs and Panels for each environment in the list
46
47         '#Region "Create Buttons"
48
49             'add this button to General Tools tab of 3 different environments
50             API_Help_Button = API_Help.CreateButton("Drawing", CustomDrawingTab, Drawing_GeneralToolsPanel)
51             API_Help_Button = API_Help.CreateButton("Assembly", CustomAssemblyTab, Assembly_GeneralToolsPanel)
52             API_Help_Button = API_Help.CreateButton("Part", CustomPartTab, Part_GeneralToolsPanel, True)
53
54             'add button to just one tab
55             Detect_Dark_Light_Theme_Button =
56                 Detect_Dark_Light_Theme.CreateButton("Drawing", CustomDrawingTab,
57
58                     Drawing_ExternalRuleButtonsPanel, True, False)
59
60             Toggle_Sheet_Color_Button =
61                 Toggle_Sheet_Color.CreateButton("Drawing", CustomDrawingTab,
62
63                     Drawing_ExternalRuleButtonsPanel, True, False)
64
65         '#End Region
66
67     End Sub
68
69     '#Region "Button 'on click' Events"
70
71     'Add button click events to this region
72
73 '#End Region
74
75 End Class
```

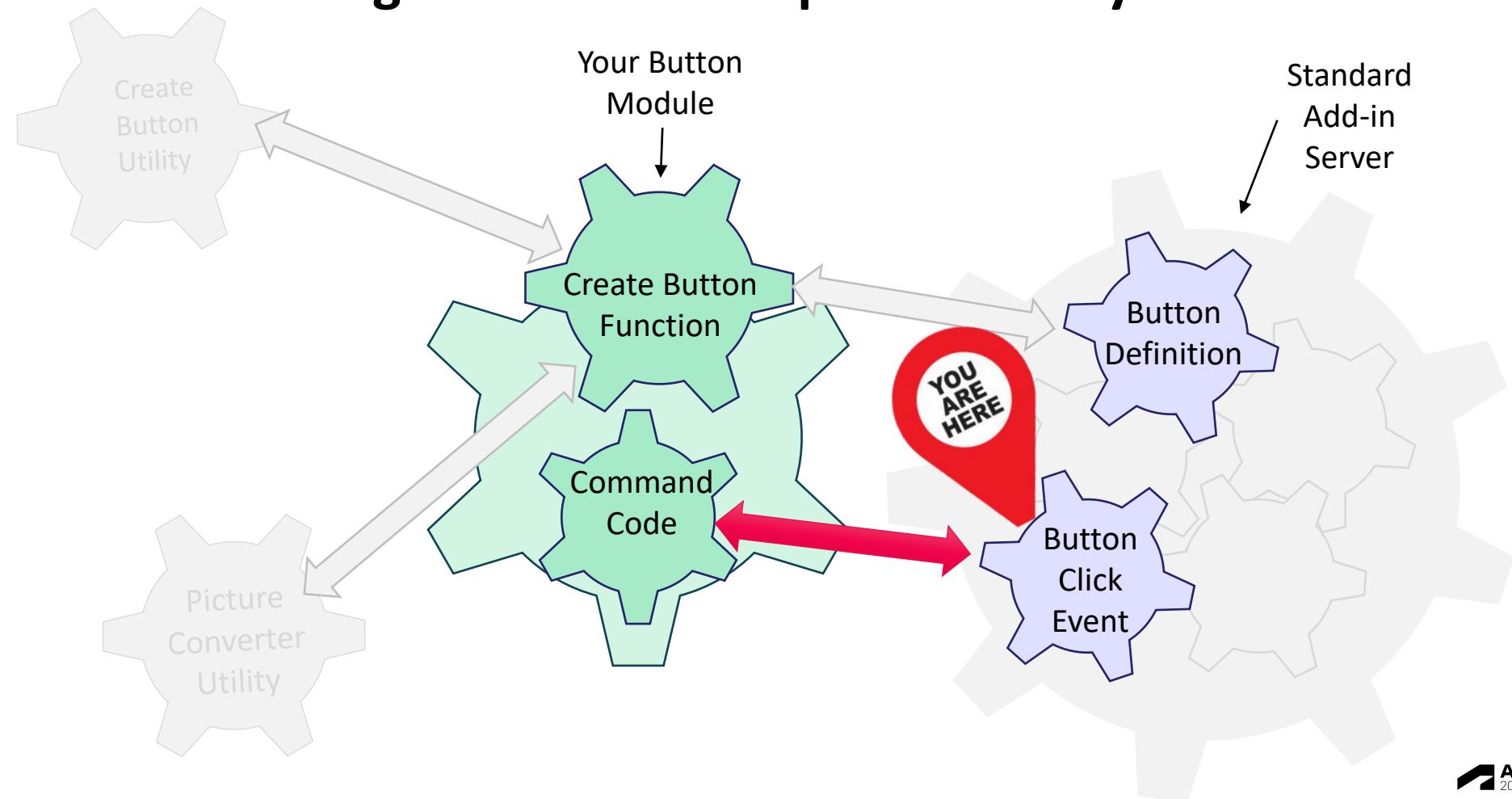
Ln: 39 Ch: 41 SPC CRLF

Ready Add to Source Control Select Repository

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

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+Q) My...ddin - □ X

StandardAddInServer.vb\* Toggle\_Sheet\_Color.vb

VB My\_New\_Addin

```
134 Detect_Dark_Light_Theme_Button =  
135     Detect_Dark_Light_Theme.CreateButton("Drawing", CustomDrawingTab,  
136                                         Drawing_ExternalRuleButtonsPanel, True, False)  
137  
138     Toggle_Sheet_Color_Button =  
139         Toggle_Sheet_Color.CreateButton("Drawing", CustomDrawingTab,  
140                                         Part_GeneralToolsPanel, True, False)  
141 #End Region  
142  
143     End Sub  
144  
145  
146     Button 'on click' Events  
147 #Region "Button 'on click' Events"  
148     'add button click events to this region  
149  
150     0 references  
151     Private Sub API_Help_Button_OnExecute(Context As NameValueMap) Handles API_Help_Button.OnExecute  
152         API_Help.RunCommandCode()  
153     End Sub  
154  
155     0 references  
156     Private Sub Detect_Dark_Light_Theme_Button_OnExecute(Context As NameValueMap) Handles Detect_Dark  
157         Detect_Dark_Light_Theme.Run_ExternalRule()  
158     End Sub  
159 #End Region  
160  
161 End Class  
162  
163 End Namespace  
164
```

S... Live Share

Search Solution 'My\_New\_A' My Project References Comma VB API\_ Help VB Togg VB Welc Resources Utility To VB Assembly Autodes

Ready

# Modifying the Add-in



```
VB My_New_Addin
144     End Sub
145
146 #Region "Button 'on click' Events"
147     'add button click events to this region
148
149     0 references
150     Private Sub API_Help_Button_OnExecute(Context As NameValueMap) Handles API_Help_Button.OnExecute
151         API_Help.RunCommandCode()
152     End Sub
153
154     0 references
155     Private Sub Detect_Dark_Light_Theme_Button_OnExecute(Context As NameValueMap) Handles Detect_Dark_Light_Theme_
156         Detect_Dark_Light_Theme.Run_ExternalRule()
157     End Sub
158
159     0 references
160     Private Sub Toggle_Sheet_Color_Button_OnExecute(Context As NameValueMap) Handles Toggle_Sheet_Color_Button.OnExecute
161         Toggle_Sheet_Color.RunCommandCode()
162     End Sub
163     #End Region
164     End Class
165
166     0 references
167     Private Sub Toggle_Sheet_Color_Button_OnExecute(Context As NameValueMap) Handles Toggle_Sheet_Color_Button.OnExecute
168         Toggle_Sheet_Color.RunCommandCode()
169     End Sub
170
171     0 references
172     Private Sub Toggle_Sheet_Color_Button_OnExecute(Context As NameValueMap) Handles Toggle_Sheet_Color_Button.OnExecute
173         Toggle_Sheet_Color.RunCommandCode()
174     End Sub
175
176     0 references
177     Private Sub Toggle_Sheet_Color_Button_OnExecute(Context As NameValueMap) Handles Toggle_Sheet_Color_Button.OnExecute
178         Toggle_Sheet_Color.RunCommandCode()
179     End Sub
180
181     0 references
182     Private Sub Toggle_Sheet_Color_Button_OnExecute(Context As NameValueMap) Handles Toggle_Sheet_Color_Button.OnExecute
183         Toggle_Sheet_Color.RunCommandCode()
184     End Sub
185
186     0 references
187     Private Sub Toggle_Sheet_Color_Button_OnExecute(Context As NameValueMap) Handles Toggle_Sheet_Color_Button.OnExecute
188         Toggle_Sheet_Color.RunCommandCode()
189     End Sub
190
191     0 references
192     Private Sub Toggle_Sheet_Color_Button_OnExecute(Context As NameValueMap) Handles Toggle_Sheet_Color_Button.OnExecute
193         Toggle_Sheet_Color.RunCommandCode()
194     End Sub
195
196     0 references
197     Private Sub Toggle_Sheet_Color_Button_OnExecute(Context As NameValueMap) Handles Toggle_Sheet_Color_Button.OnExecute
198         Toggle_Sheet_Color.RunCommandCode()
199     End Sub
200
201     0 references
202     Private Sub Toggle_Sheet_Color_Button_OnExecute(Context As NameValueMap) Handles Toggle_Sheet_Color_Button.OnExecute
203         Toggle_Sheet_Color.RunCommandCode()
204     End Sub
205
206     0 references
207     Private Sub Toggle_Sheet_Color_Button_OnExecute(Context As NameValueMap) Handles Toggle_Sheet_Color_Button.OnExecute
208         Toggle_Sheet_Color.RunCommandCode()
209     End Sub
210
211     0 references
212     Private Sub Toggle_Sheet_Color_Button_OnExecute(Context As NameValueMap) Handles Toggle_Sheet_Color_Button.OnExecute
213         Toggle_Sheet_Color.RunCommandCode()
214     End Sub
215
216     0 references
217     Private Sub Toggle_Sheet_Color_Button_OnExecute(Context As NameValueMap) Handles Toggle_Sheet_Color_Button.OnExecute
218         Toggle_Sheet_Color.RunCommandCode()
219     End Sub
220
221     0 references
222     Private Sub Toggle_Sheet_Color_Button_OnExecute(Context As NameValueMap) Handles Toggle_Sheet_Color_Button.OnExecute
223         Toggle_Sheet_Color.RunCommandCode()
224     End Sub
225
226     0 references
227     Private Sub Toggle_Sheet_Color_Button_OnExecute(Context As NameValueMap) Handles Toggle_Sheet_Color_Button.OnExecute
228         Toggle_Sheet_Color.RunCommandCode()
229     End Sub
230
231     0 references
232     Private Sub Toggle_Sheet_Color_Button_OnExecute(Context As NameValueMap) Handles Toggle_Sheet_Color_Button.OnExecute
233         Toggle_Sheet_Color.RunCommandCode()
234     End Sub
235
236     0 references
237     Private Sub Toggle_Sheet_Color_Button_OnExecute(Context As NameValueMap) Handles Toggle_Sheet_Color_Button.OnExecute
238         Toggle_Sheet_Color.RunCommandCode()
239     End Sub
240
241     0 references
242     Private Sub Toggle_Sheet_Color_Button_OnExecute(Context As NameValueMap) Handles Toggle_Sheet_Color_Button.OnExecute
243         Toggle_Sheet_Color.RunCommandCode()
244     End Sub
245
246     0 references
247     Private Sub Toggle_Sheet_Color_Button_OnExecute(Context As NameValueMap) Handles Toggle_Sheet_Color_Button.OnExecute
248         Toggle_Sheet_Color.RunCommandCode()
249     End Sub
250
251     0 references
252     Private Sub Toggle_Sheet_Color_Button_OnExecute(Context As NameValueMap) Handles Toggle_Sheet_Color_Button.OnExecute
253         Toggle_Sheet_Color.RunCommandCode()
254     End Sub
255
256     0 references
257     Private Sub Toggle_Sheet_Color_Button_OnExecute(Context As NameValueMap) Handles Toggle_Sheet_Color_Button.OnExecute
258         Toggle_Sheet_Color.RunCommandCode()
259     End Sub
260
261     0 references
262     Private Sub Toggle_Sheet_Color_Button_OnExecute(Context As NameValueMap) Handles Toggle_Sheet_Color_Button.OnExecute
263         Toggle_Sheet_Color.RunCommandCode()
264     End Sub
265
266     0 references
267     Private Sub Toggle_Sheet_Color_Button_OnExecute(Context As NameValueMap) Handles Toggle_Sheet_Color_Button.OnExecute
268         Toggle_Sheet_Color.RunCommandCode()
269     End Sub
270
271     0 references
272     Private Sub Toggle_Sheet_Color_Button_OnExecute(Context As NameValueMap) Handles Toggle_Sheet_Color_Button.OnExecute
273         Toggle_Sheet_Color.RunCommandCode()
274     End Sub
275
276     0 references
277     Private Sub Toggle_Sheet_Color_Button_OnExecute(Context As NameValueMap) Handles Toggle_Sheet_Color_Button.OnExecute
278         Toggle_Sheet_Color.RunCommandCode()
279     End Sub
280
281     0 references
282     Private Sub Toggle_Sheet_Color_Button_OnExecute(Context As NameValueMap) Handles Toggle_Sheet_Color_Button.OnExecute
283         Toggle_Sheet_Color.RunCommandCode()
284     End Sub
285
286     0 references
287     Private Sub Toggle_Sheet_Color_Button_OnExecute(Context As NameValueMap) Handles Toggle_Sheet_Color_Button.OnExecute
288         Toggle_Sheet_Color.RunCommandCode()
289     End Sub
290
291     0 references
292     Private Sub Toggle_Sheet_Color_Button_OnExecute(Context As NameValueMap) Handles Toggle_Sheet_Color_Button.OnExecute
293         Toggle_Sheet_Color.RunCommandCode()
294     End Sub
295
296     0 references
297     Private Sub Toggle_Sheet_Color_Button_OnExecute(Context As NameValueMap) Handles Toggle_Sheet_Color_Button.OnExecute
298         Toggle_Sheet_Color.RunCommandCode()
299     End Sub
299
```

- Add a sub to run the Toggle\_Sheet\_Color Module command code when the button is executed

# Modifying the Add-in



StandardAddInServer.vb\* Toggle\_Sheet\_Color.vb

```
VB My_New_Addin
144
145
146
147 End Sub
148 #Region "Button 'on click' Events"
149 'add button click events to this region
150
151 Private Sub API_Help_Button_OnExecute(Context As NameValueMap) Handles API_Help.Button.OnExecute
152     API_Help.RunCommandCode()
153 End Sub
154
155 #Region "Detect Dark/Light Theme Button Events"
156 Private Sub Detect_Dark_Light_Theme_Button_OnExecute(Context As NameValueMap) Handles Detect_Dark_Light_Theme.Button.OnExecute
157     Detect_Dark_Light_Theme.Run_ExternalRule()
158 End Sub
159 #End Region
160
161 Private Sub Toggle_Sheet_Color_Button_OnExecute(Context As NameValueMap) Handles Toggle_Sheet_Color_Button.OnExecute
162     Toggle_Sheet_Color.Run_ExternalRule()
163 End Sub
164
165 End Class
```

0 references

```
Private Sub Toggle_Sheet_Color_Button_OnExecute(Context As NameValueMap) Handles Toggle_Sheet_Color_Button.OnExecute
    Toggle_Sheet_Color.Run_ExternalRule()
End Sub
```

100% No issues found | ↻ Ln: 312 Ch: 1 SPC CRLF | ↑ Add to Source Control | Select Repository | Ready

Alternative iLogic Rule workflow!

You could make this show up on the external rules panel

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

# Overall Process



Visual  
Studio



- ✓ Write
- ✓ Debug
- ✓ Build

```
1 // Text that shows when the user
2 // has selected a single line or multiple lines, based on ca
3 // set to true to use a progressive tool tip, and false to a simple tool tip
4 // Set to true if user requested tool tip = true
5 // Only used if user requested tool tip = true
6 // User requested tool tip = true
7 // Changes the edges in all views in all views on all
8 // Ch(140) & * Edges are pulled from the model col.
9 // Ch(140) & * Edges are pulled from the model col.
10 // Ch(140) & * Edges are pulled from the model col.
11 // Ch(140) & * Edges are pulled from the model col.
12 // Ch(140) & * Edges are pulled from the model col.
13 // Ch(140) & * Edges are pulled from the model col.
14 // Ch(140) & * Edges are pulled from the model col.
15 // Ch(140) & * Edges are pulled from the model col.
16 // Ch(140) & * Edges are pulled from the model col.
17 // Ch(140) & * Edges are pulled from the model col.
18 // Ch(140) & * Edges are pulled from the model col.
19 // Ch(140) & * Edges are pulled from the model col.
20 // Ch(140) & * Edges are pulled from the model col.
21 // Ch(140) & * Edges are pulled from the model col.
22 // Ch(140) & * Edges are pulled from the model col.
```

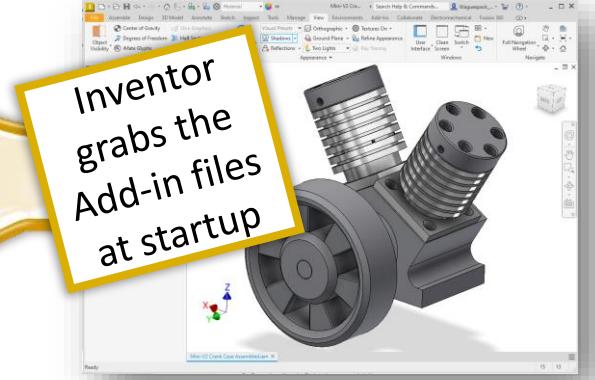
VS compiles the code and writes out the DLL and ADDIN files



Autodesk  
Application Plugins  
Folder



Autodesk  
Inventor



# Compiling the Add-in



The screenshot shows the Microsoft Visual Studio interface. In the center is the Solution Explorer window, which lists a single project named "My\_New\_Addin". A context menu is open over the project node, with the "Build Solution" option highlighted with a red box. To the left is the code editor showing VBA code for "StandardAddInServer.vb". The code includes regions for button click events and application events. At the bottom is the Output window displaying build logs.

```
StandardAddInServer.vb* Toggle_Sheet_Color.vb
VB My_New_Addin
144 End Sub
145
146 '#Region "Button 'on click' Events"
147 'add button click events to this region
148
149 0 references
150 Private Sub API_Help_Button_OnExecute(Context As
151     API_Help.RunCommandCode()
152 End Sub
153
154 0 references
155 Private Sub Detect_Dark_Light_Theme_Button_OnExe
156     Detect_Dark_Light_Theme.Run_ExternalRule()
157 End Sub
158 '#End Region
159
160 #Application Events
161
162 #ApplicationAddInServer Members
163
164 End Class
Output
Show output from: Build
1> INFO: Could not find files for the given pattern(s).
1> The system cannot find the path specified.
1> 'mt.exe' is not recognized as an internal or external command,
1> operable program or batch file.
1> D:\Autodesk University\AU 2023\Add Ins\My_New_Add-In\bin\Debug\My_New_A
1> 1 File(s) copied
1> D:\Autodesk University\AU 2023\Add Ins\My_New_Add-In\Autodesk.My_New_Ad
1> 1 File(s) copied
===== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ======
===== Build started at 4:25 PM and took 00.680 seconds ======
Error List Output
Build succeeded
Ready
```

- The solution is shown in the Solution Explorer



# Compiling the Add-in



The screenshot shows the Microsoft Visual Studio interface. The menu bar is visible at the top, with the 'Build' menu highlighted. A red arrow points from the text in the yellow sticky note to the 'Output' option in the 'Build' menu. The 'Output' tab is currently selected in the bottom-left 'Output' window. The main 'Solution Explorer' window shows a project named 'My\_New\_AddIn'. A yellow sticky note with handwritten text is overlaid on the interface, pointing towards the 'Output' tab.

Tip: you can turn the Output window on/off using the View tab

Output

```
Show output from: Build
1> INFO: Could not find files for the given pattern(s).
1> The system cannot find the path specified.
1: D:\Autodesk University\AU 2023\Add Ins\My_New_AddIn\bin\Debug\My_New_AddIn.dll
1 File(s) copied
1: D:\Autodesk University\AU 2023\Add Ins\My_New_AddIn\Autodesk.My_New_AddIn.Inventor.addin
1 File(s) copied
===== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped =====
```

Build succeeded

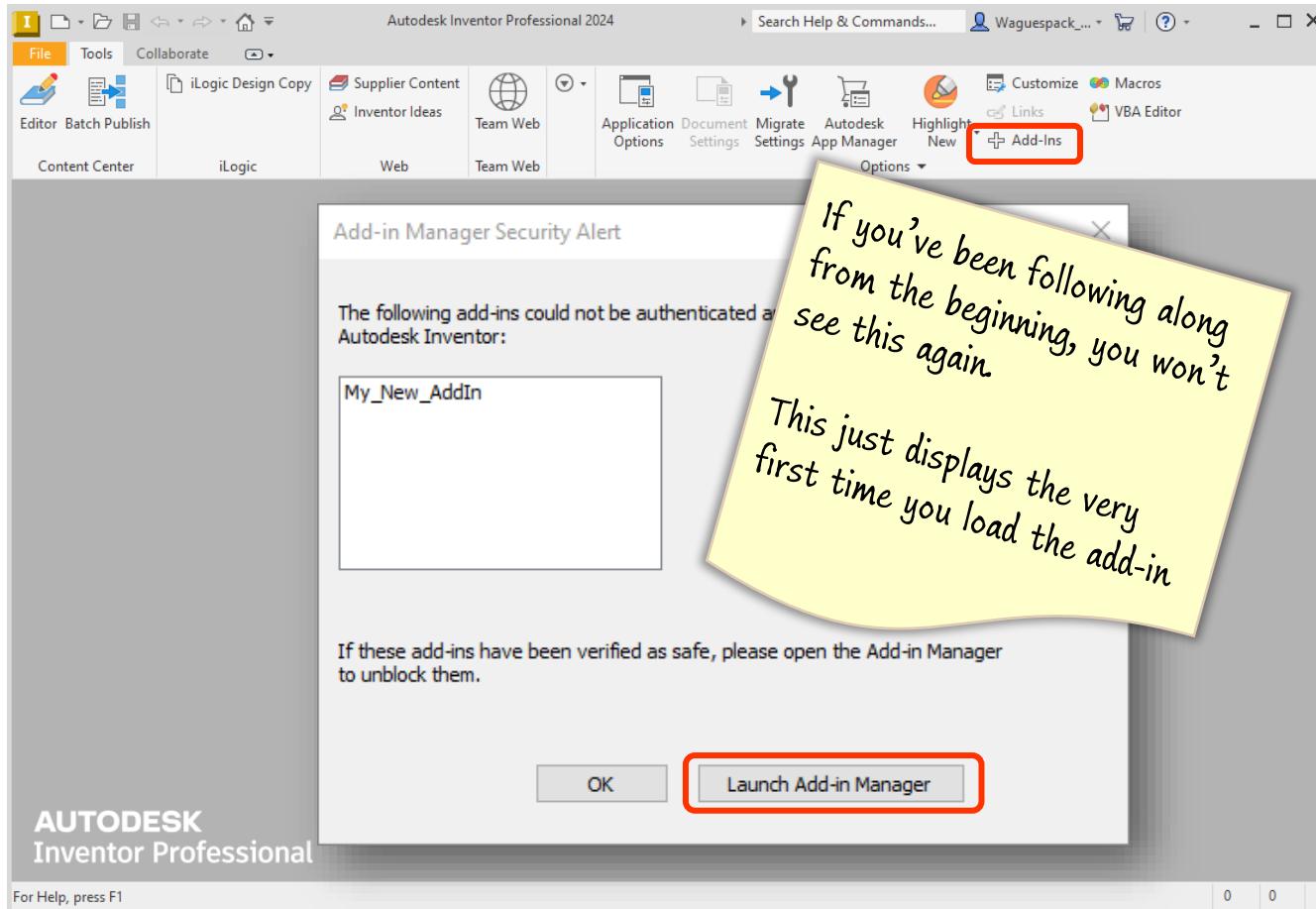
- The Build results are shown in the Output window

# Testing the add-in

Working in Inventor

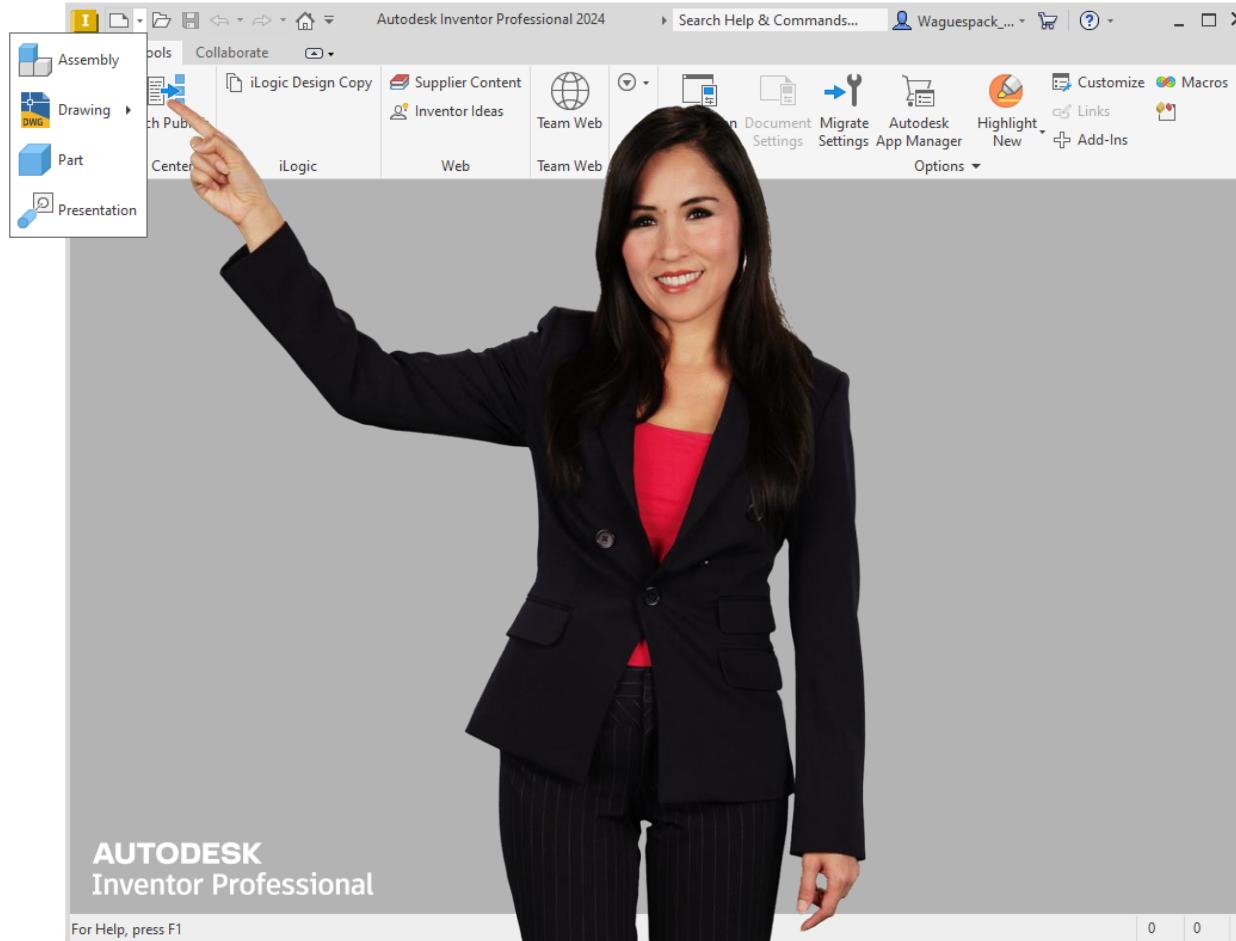


# Testing the Add-in



- 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 go to Tools > Add-ins

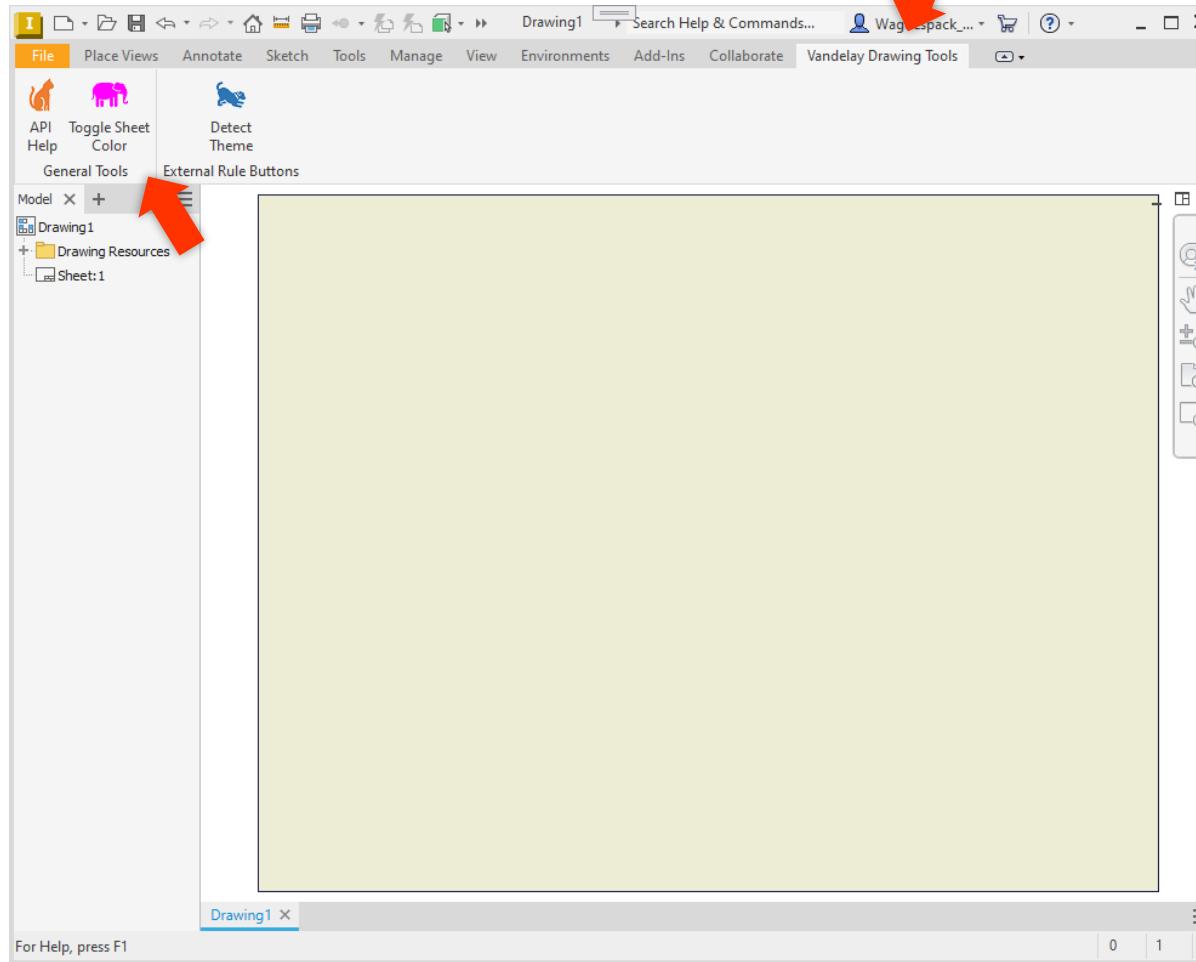
# Using the Add-in



- Create a new drawing file from a template

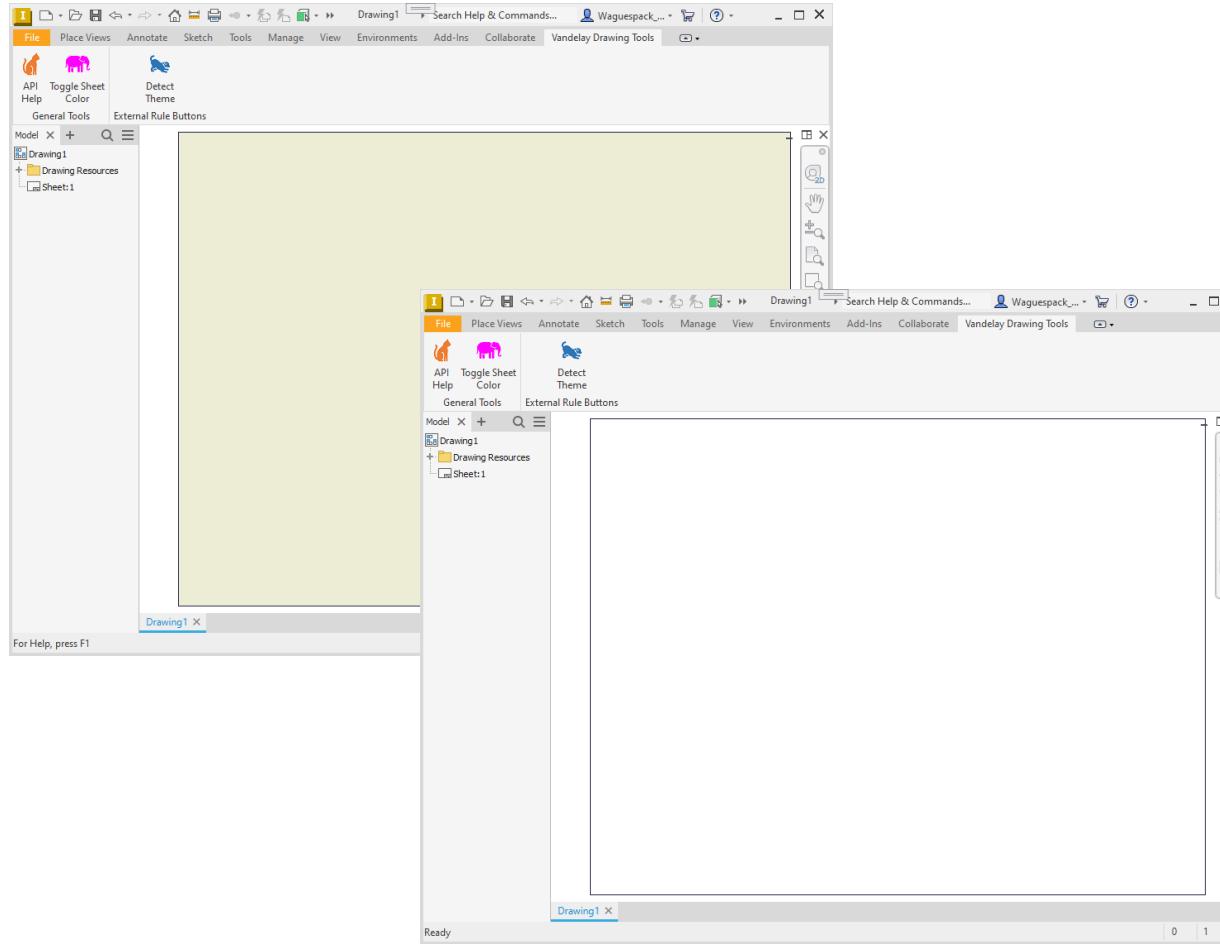
AUTODESK  
Inventor Professional

# Using the Add-in



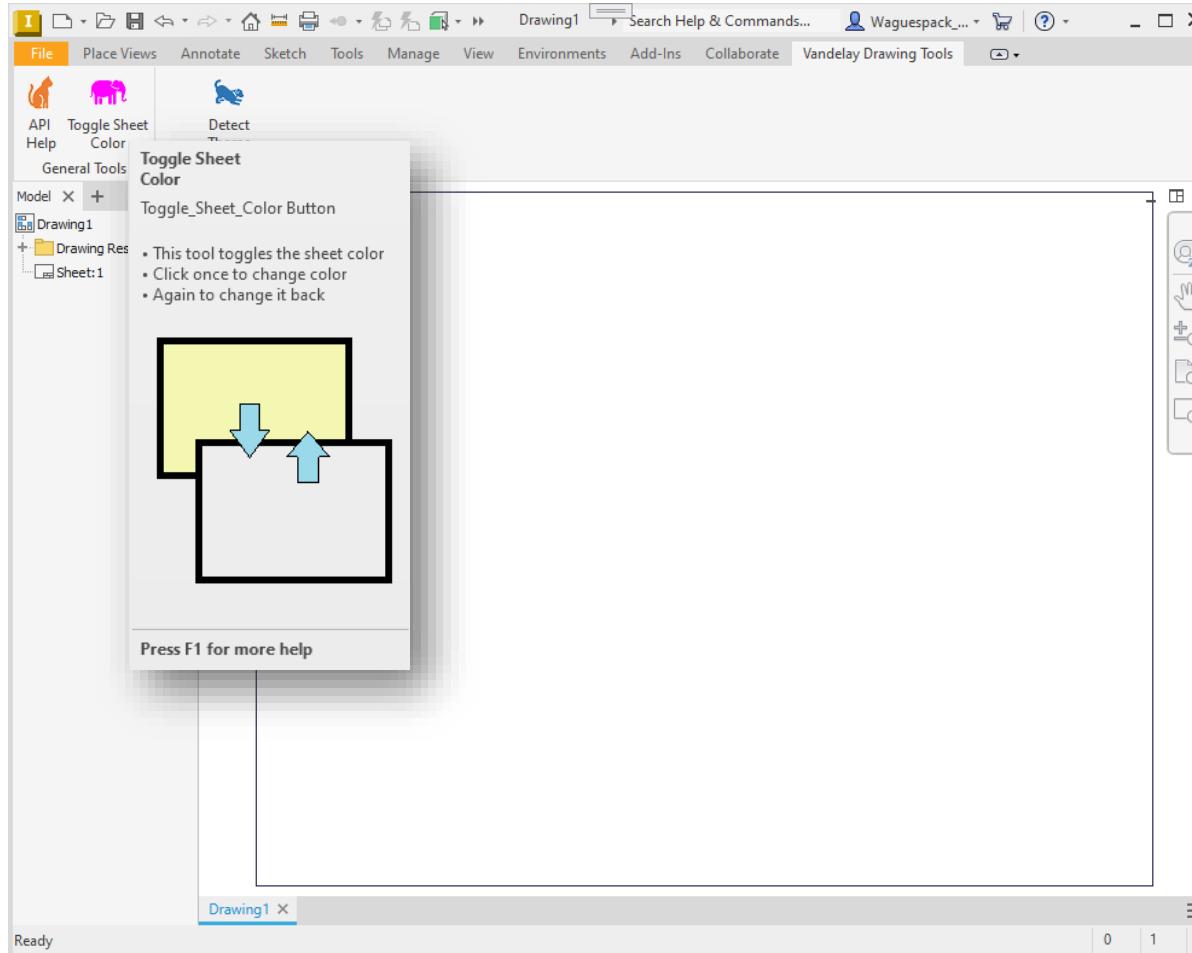
- Switch to the **Vandelay Drawing Tool** tab
- Note the new button on this tab

# 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

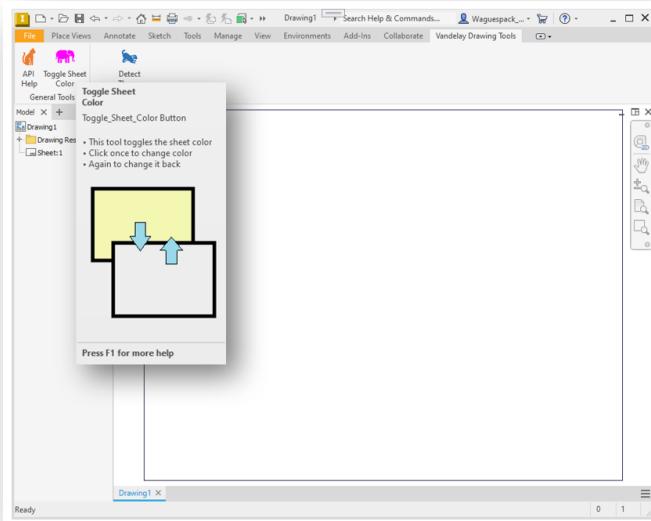
# Using the Add-in



- 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!

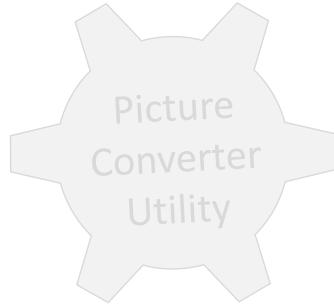
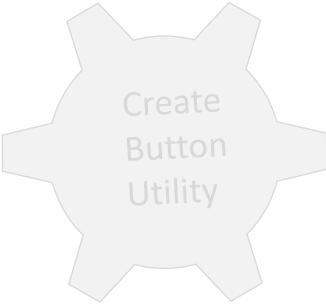


# Quick Review

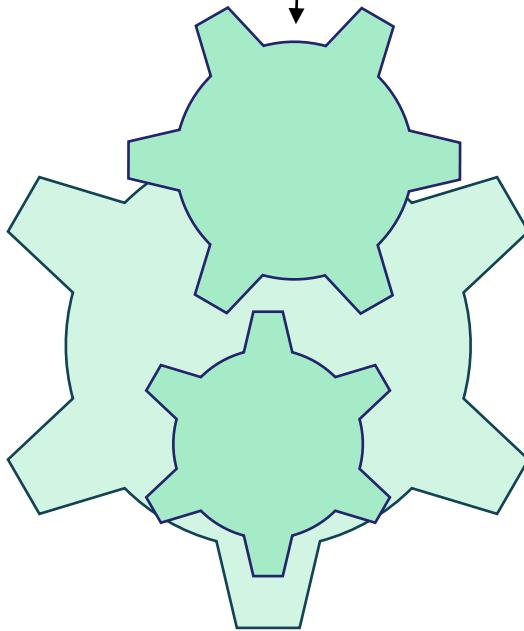
- Copy an existing Command Module to create a new one
- Open the module and change the module name
- In the CreateButton function:
  - Add button icons to the Resources
  - Set the button icons
  - Set the button label name
  - Set the simple tooltip
  - Set the progressive tooltip if desired
- In the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

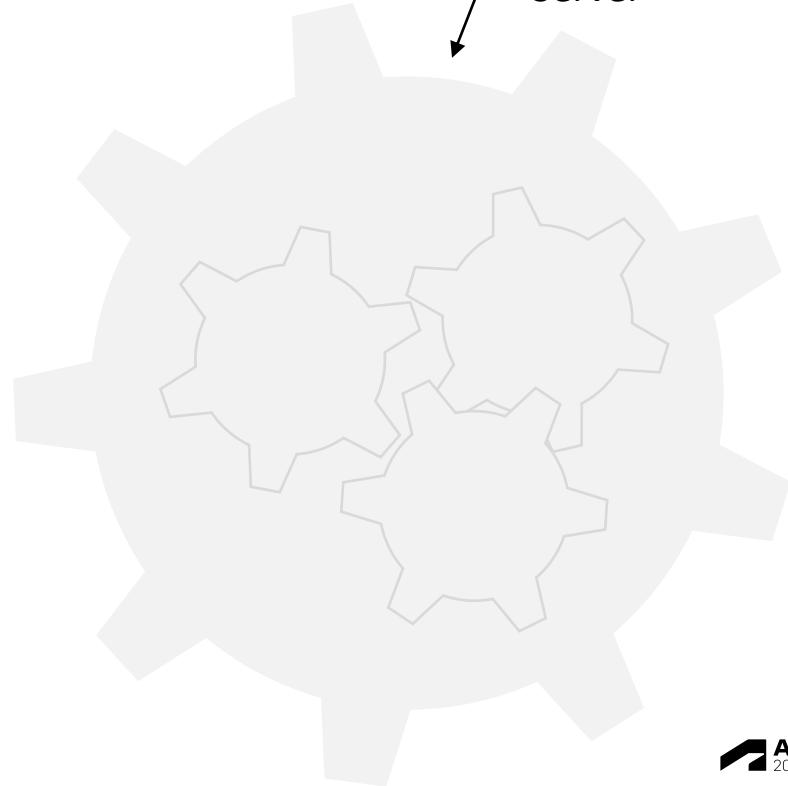
# Review



Copy/Paste to create a new button module



Standard Add-in Server



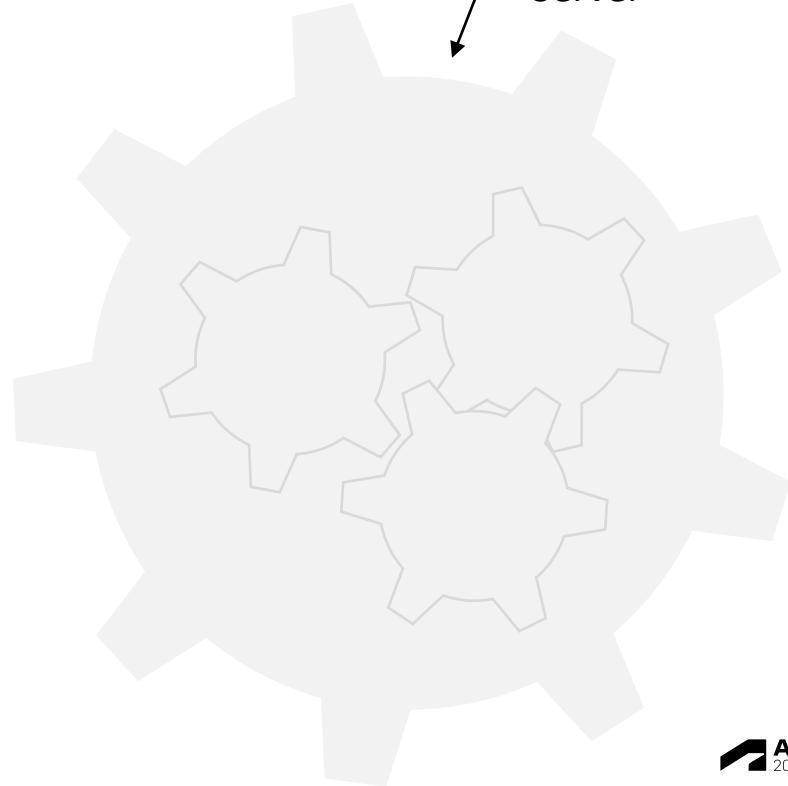
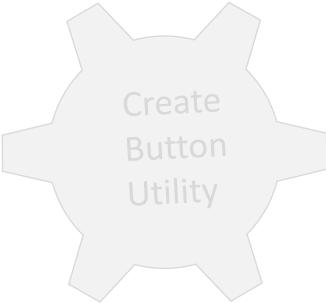
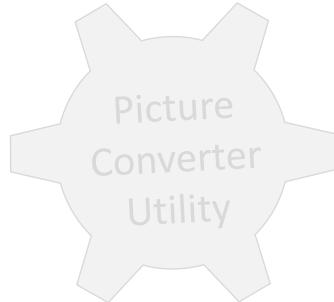
# Review

Copy/Paste to create a new button module

Button name  
Button icons  
Button tooltip

Create Button Function

Standard Add-in Server



# Review

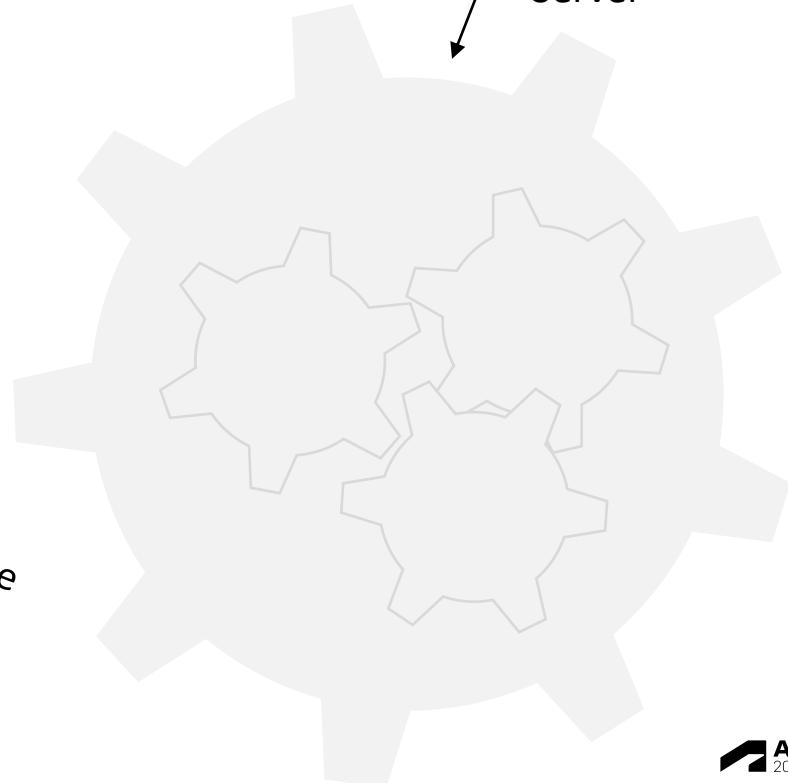
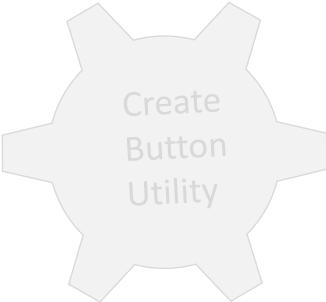
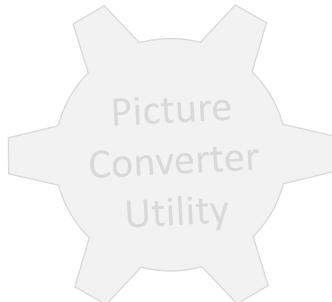
Copy/Paste to create a new button module

Create Button Function

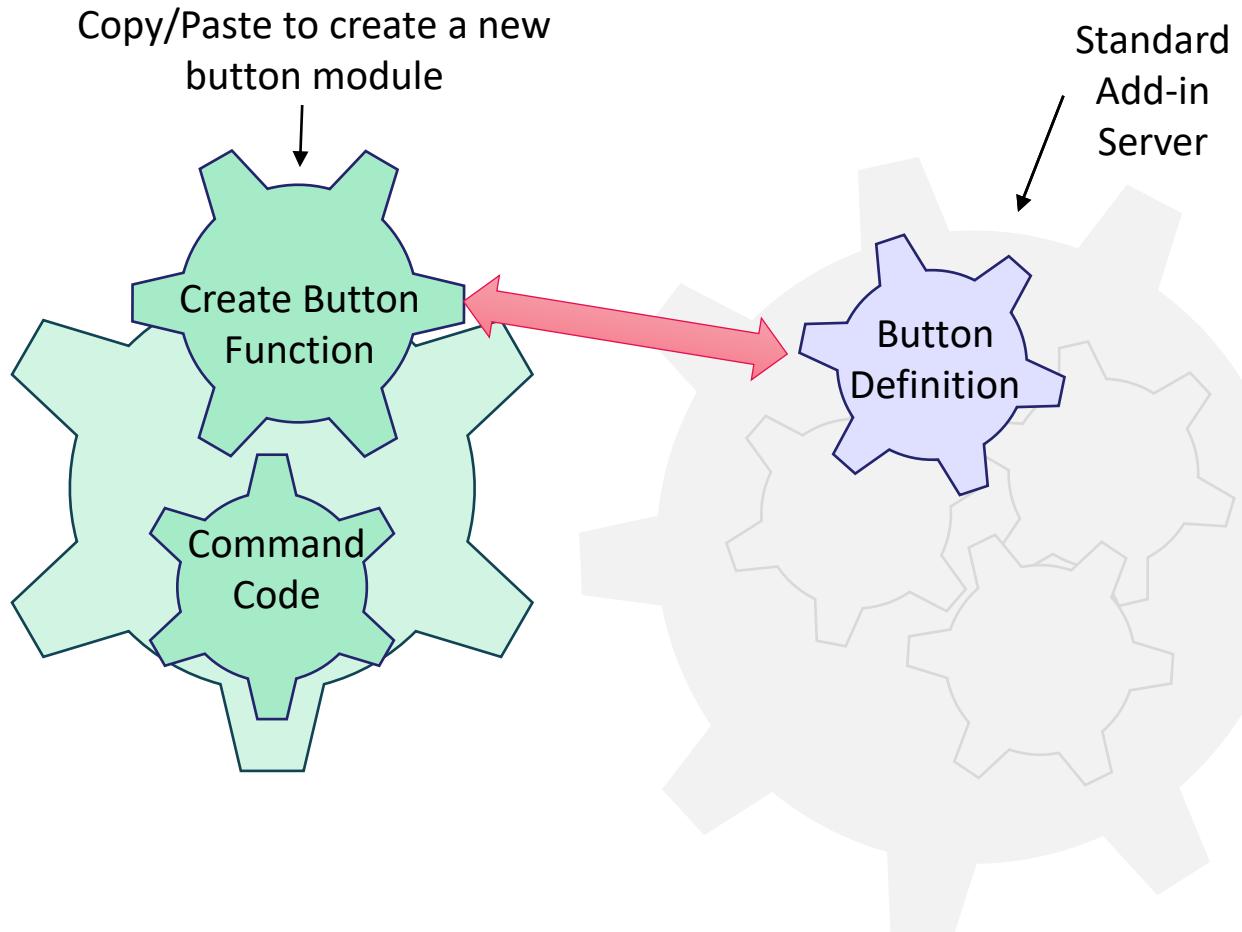
Command Code

Code to be  
executed  
when the  
button is  
clicked

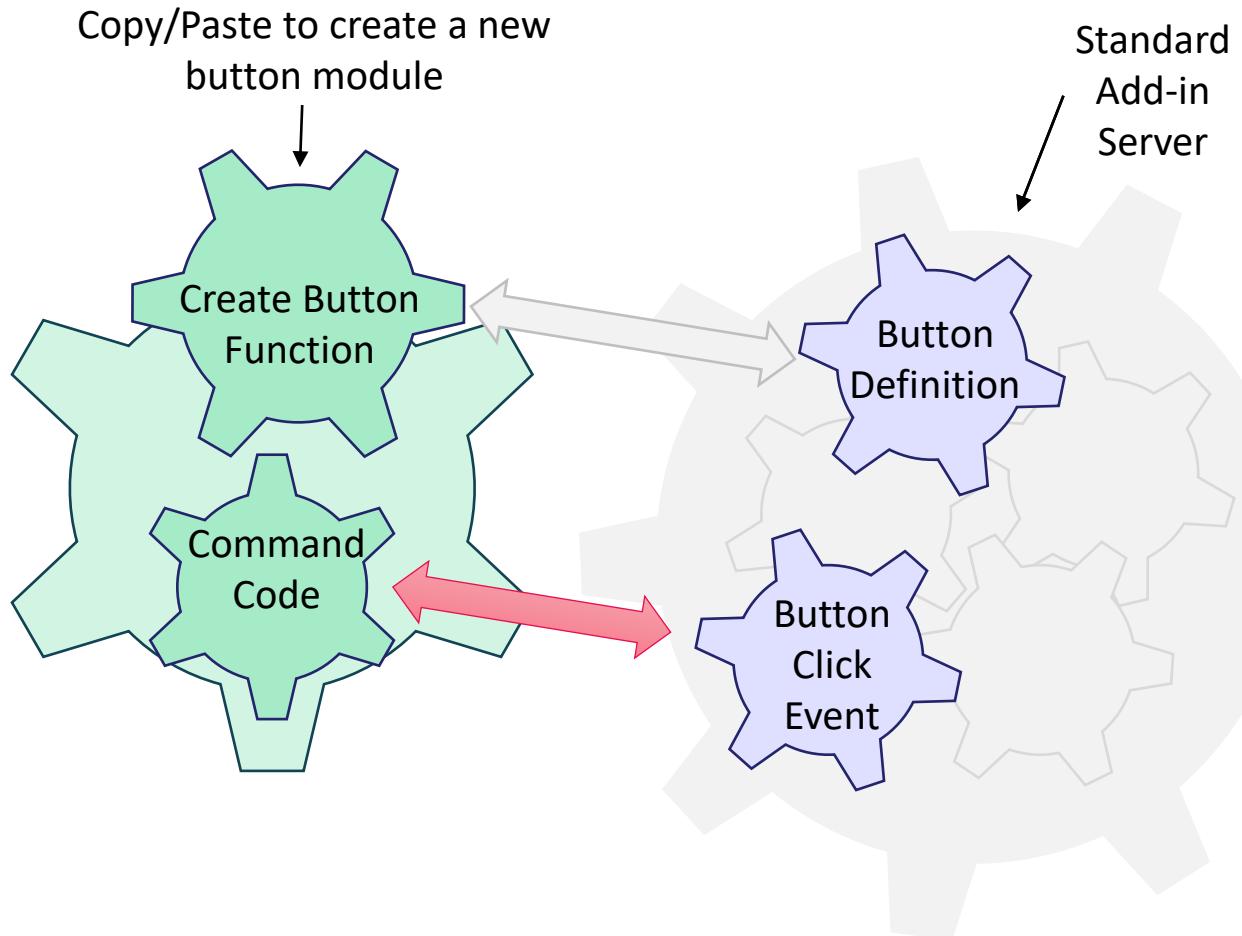
Standard  
Add-in  
Server



# Review

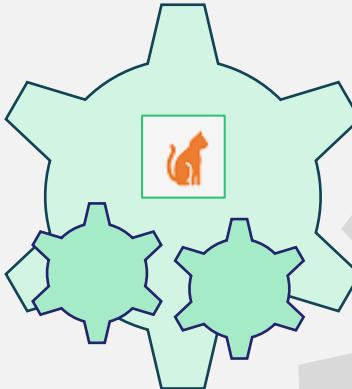


# Review

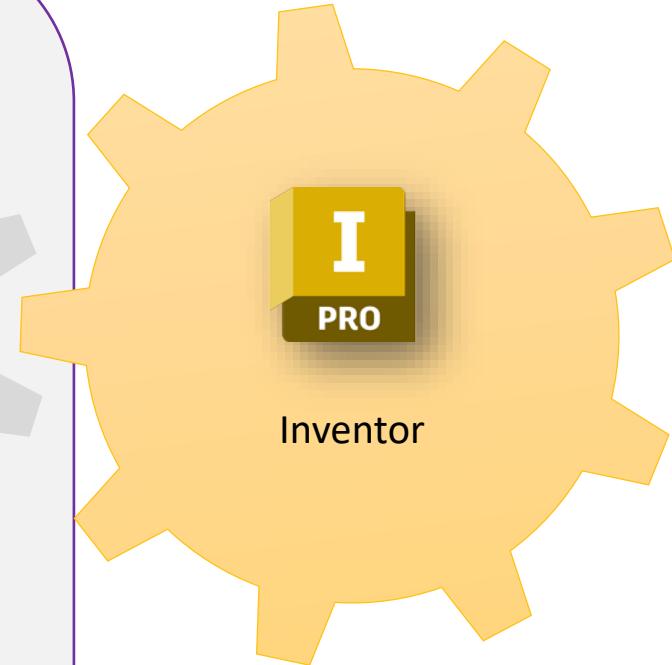
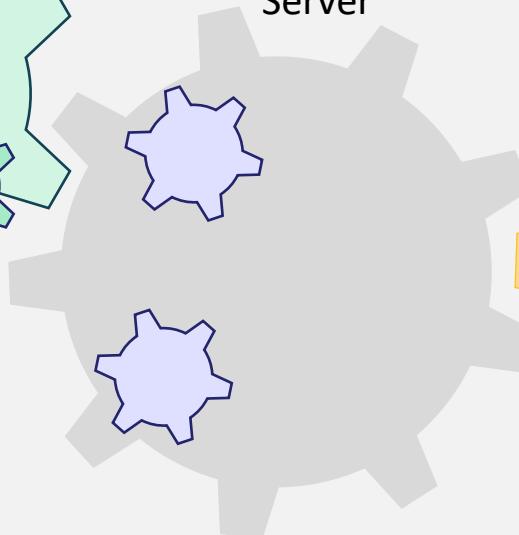


# Review

Utilities

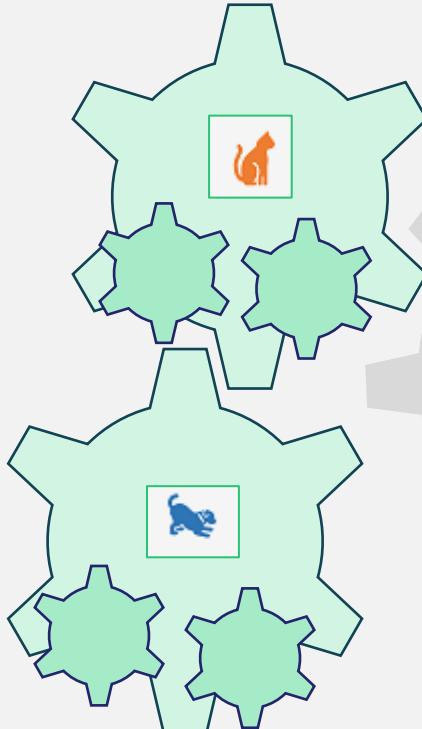


Standard Add-in  
Server

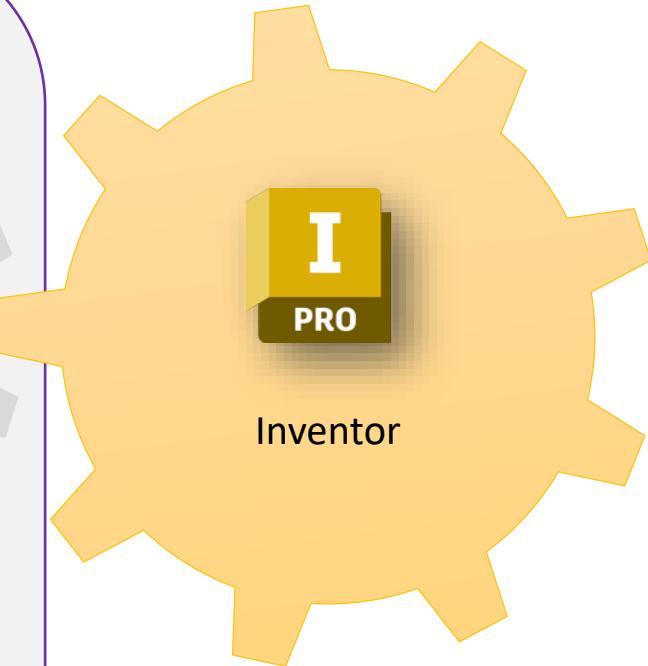
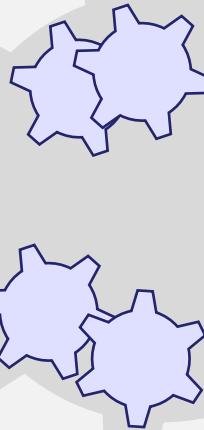


# Review

Utilities



Standard Add-in  
Server



# Understanding the Add-in Template Structure

Utilities

Standard Add-in  
Server

Inventor

Thank you!



THE DESIGN & MAKE CONFERENCE