apper

Release .1

Patrick Rainsberry

May 27, 2021

CONTENTS

1 The User Guide **3**
- 1.1 Intro . . . 3
- 1.2 Installation . . . 3
- 1.3 Existing Projects . . . 6
- 1.4 3rd Party Libraries . . . 7
- 1.5 Commands . . . 8
- 1.6 Events . . . 12

2 The API Documentation / Guide **13**
- 2.1 Developer Interface . . . 13

3 Indices and tables **23**

Python Module Index **25**

Index **27**

Apper is a framework to simplify the creation of Fusion 360 Addin.

There are many tasks and to launch and clean up a standard Fusion 360 add-in. This project aims to simplify that process and help you get started as quickly as possible.

It also includes a number of resources to simplify and speed up the process of creating Fusion 360 add-ins.

CHAPTER

ONE

THE USER GUIDE

This part of the documentation, will give you a quick introduction to the project and help get you started creating your first add-in.

1.1 Intro

TODO

Will add some description of working with the Fusion 360 API and the rational for the project

1.2 Installation

The easiest way to get started with apper is to start from a template project.

This will download and structure a new add-in for you on your local system.

You can set some basic parameters and the template will generate everything you need to get started.

1.2.1 Prerequisites

- Python interpreter
- Install Git
- Adjust your path
- Packaging tools

Python interpreter

Install Python for your operating system. Fusion 360 uses Python 3.7 so it is recommended to install this version locally as it will simplify setting up your development environment in general.

Consult the official Python documentation for details.

You can install the Python binaries from python.org.

Install Git

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

You will need to have git installed to properly setup your local environment. It is recomended to just install github desktop if you do not already have git installed locally.

Alternatively you can review other installation options.

Adjust your path

Ensure that your `bin` folder is on your path for your platform. Typically `~/.local/` for UNIX and macOS, or `%APPDATA%\Python` on Windows. (See the Python documentation for site.USER_BASE for full details.)

MacOS

For bash shells, add the following to your `.bash_profile` (adjust for other shells):

```
# Add ~/.local/ to PATH
export PATH=$HOME/.local/bin:$PATH
```

Remember to load changes with `source ~/.bash_profile` or open a new shell session.

Windows

Ensure the directory where cookiecutter will be installed is in your environment's `Path` in order to make it possible to invoke it from a command prompt. To do so, search for "Environment Variables" on your computer (on Windows 10, it is under `System Properties` –> `Advanced`) and add that directory to the `Path` environment variable, using the GUI to edit path segments.

Example segments should look like `%APPDATA%\Python\Python3x\Scripts`, where you have your version of Python instead of `Python3x`.

You may need to restart your command prompt session to load the environment variables.

See also:

See Configuring Python (on Windows) for full details.

Install cookiecutter

cookiecutter creates projects from project templates and is an amazing resource

For more detailed installation instructions see their documentation

First install cookie cutter into your local python environment

```
pip install cookiecutter
```

Or potentially if you have a separate python 3 installation you may need to use:

```
pip3 install cookiecutter
```

1.2.2 Using the Template

Navigate to the Fusion 360 Addins directory

Putting your addin in the following directory will allow Fusion 360 to automatically recognize it

Mac:

```
cd ~
cd /Library/Application Support/Autodesk/Autodesk\ Fusion\ 360/API/AddIns/
```

Windows:

```
cd  C:\Users\%YOUR_USER_NAME%\AppData\Roaming\Autodesk\Autodesk Fusion 360\API\AddIns
```

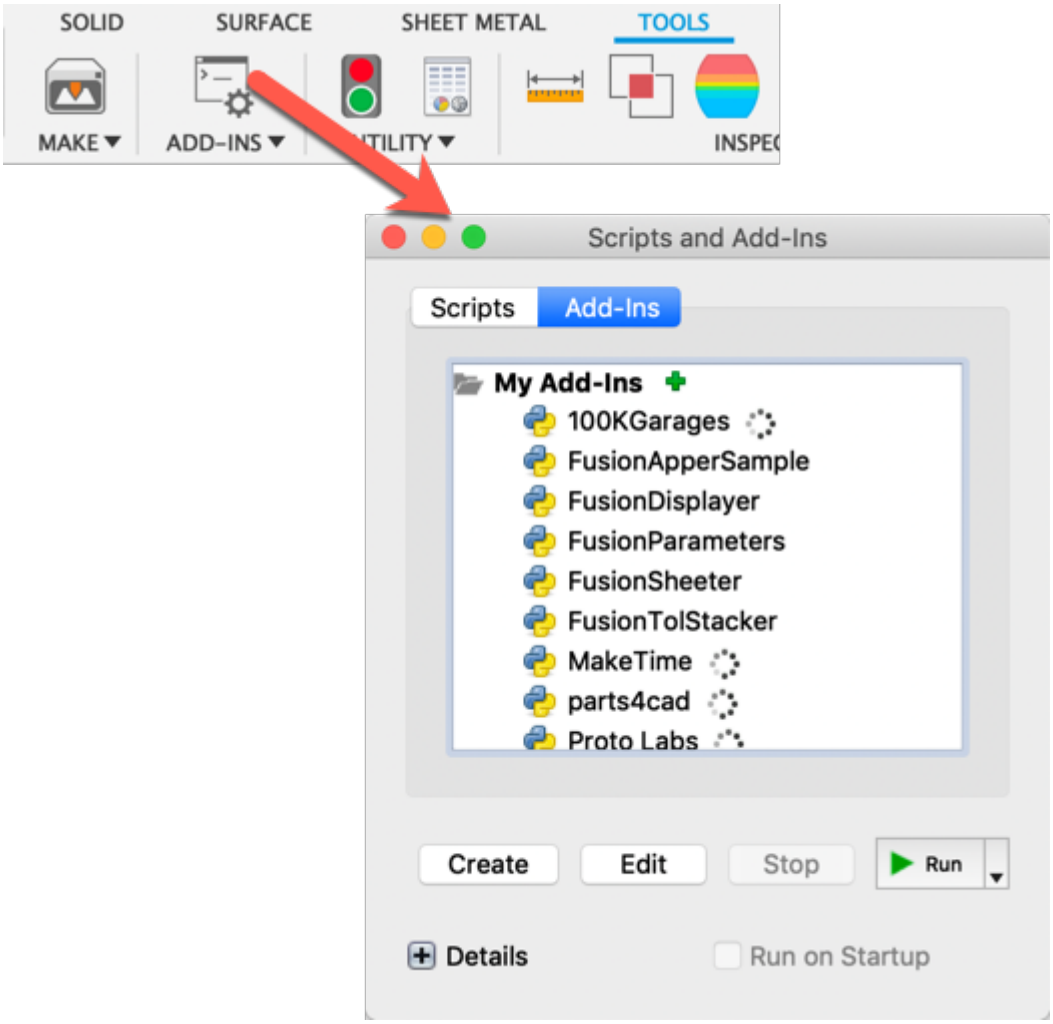
Run the cookiecutter template

This will create your add-in directory.

```
cookiecutter https://github.com/tapnair/cookiecutter-fusion360-addin.git
```

Open your new add-in

In Fusion 360 click on the tools tab and select the **Scripts and Add-ins** command



You can now either **Run** your new add-in or select **Edit** to open it in VS Code

1.3 Existing Projects

Adding apper to an existing project is not too difficult

1.3.1 Adding a Git Submodule

The best way to leverage apper in your addin project is to use Git Submodules

This way you can easily update to the latest version of apper if it is enhanced

Note: if you are using the Template files from HERE then this step is already done for you

This assumes you already have your project in a Git Repository

Open a terminal and navigate to your addin's root directory:

You should be someplace like this:

```
$ pwd

/Users/rainsbp/Library/Application Support/Autodesk/Autodesk Fusion 360/API/AddIns/
↪FusionApperSample
```

Now add the submodule to your project:

```
$ git submodule add https://github.com/tapnair/apper
Cloning into '/Users/rainsbp/Library/Application Support/Autodesk/Autodesk Fusion 360/
↪API/AddIns/FusionApperSample/apper'...
remote: Enumerating objects: 31, done.
remote: Counting objects: 100% (31/31), done.
remote: Compressing objects: 100% (25/25), done.
remote: Total 31 (delta 6), reused 29 (delta 4), pack-reused 0
Unpacking objects: 100% (31/31), done.
```

1.3.2 Status of a Git Submodule

To check the status of apper from the project root directory:

```
$ git submodule status
+e951ad1030b6ed8fb60db3bac7e1098d64289833 apper (remotes/origin/HEAD)
```

1.3.3 Update a Git Submodule

As apper continues to be developed, the advantage of submodules is that you can always simply and easily updated the apper framework inside of your addin.

To update apper from the project root directory:

```
$ git submodule update --remote
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 1), reused 4 (delta 1), pack-reused 0
Unpacking objects: 100% (4/4), done.
From https://github.com/tapnair/apper 5035ffb..e951ad1  master     -> origin/master
Submodule path 'apper': checked out 'e951ad1030b6ed8fb60db3bac7e1098d64289833'
```

1.4 3rd Party Libraries

There is an included helper class to use 3rd party libraries in a reasonably “safe” way.

1.4.1 Using 3rd Party Libraries with Fusion 360 Add-ins

Because Fusion 360 uses its own internal python runtime for the execution of add-ins there are some unique challenges to using 3rd party libraries.

Especially when those libraries have dependancies on other additional libraries. For example, Requests actually requires a number of other libraries. These libraries are expecting each other to also be in the sys.path of the currently running python interpreter. So it is not sufficient to simply install Requests to a project subdirectory and use a relative import, since even though you have imported requests, modules within requests will attempt to directly import other modules that requests installed as dependencies.

Here is one method that can be used to deal with these issues:

1. Install the library to a subdirectory of your project such as 'lib'
2. Temporarily add the location of that directory to your system path
3. Import the required package
4. Use the package
5. Remove the location from the system path

1.4.2 The lib_import class

There is a decorator class in *Fusion360Utilities* called: *lib_import* that will simplify this process for you.

1.5 Commands

1.5.1 App Structure

Once you have executed the cookiecutter template. You will have the following directory and file structure in your new addin folder.

When running an addin Fusion 360 expects to see a directory with a .py and .manifest file all with the same name. This is the minimum requirement for your application to be recognized. You should see these two files with your app name in the new directory. The manifest file doesn't really require much editing.

```
ApperSample
├── apper
│   └── ...
├── commands
│   ├── __init__.py
│   ├── SampleActiveSelectionEvents.py
│   ├── SampleCommand1.py
│   ├── SampleCommand2.py
│   ├── SampleCommandEvents.py
│   ├── SampleCustomEvent.py
│   ├── SampleDocumentEvents.py
│   ├── SamplePaletteCommand.py
│   ├── SampleWebRequestEvent.py
│   ├── SampleWorkspaceEvent.py
│   ├── palette_html
│   │   └── ApperSample.html
│   └── resources
│       ├── command_icons
```

(continues on next page)

(continued from previous page)

```
│       │   └── ...
│       └── palette_icons
│           └── ...
├── lib
│   └── Place_3rd_Party_libraries_here.txt
├── ApperSample.py
├── ApperSample.manifest
├── config.py
├── startup.py
├── LICENCE
└── README.md
```

Your_App.py (ApperSample in this case) is the main entry point to the app. Here you will define the commands that will be added and where they will be placed in the ui.

1.5.2 Imports

In this sample the commands and events are defined in a number of files that need to be imported. Typically I create each command in its own file unless there are two commands that will be sharing much of the same logic, but it doesn't really matter.

```
import adsk.core
import traceback

from.startup import setup_app, cleanup_app, get_app_path
setup_app(__file__)

try:
    import config
    import apper

    # Basic Fusion 360 Command Base samples
    from commands.SampleCommand1 import SampleCommand1
    from commands.SampleCommand2 import SampleCommand2

    # Palette Command Base samples
    from commands.SamplePaletteCommand import SamplePaletteSendCommand,␣
↪SamplePaletteShowCommand

    # Various Application event samples
    from commands.SampleCustomEvent import SampleCustomEvent
    from commands.SampleDocumentEvents import SampleDocumentEvent1,␣
↪SampleDocumentEvent2
    from commands.SampleWorkspaceEvents import SampleWorkspaceEvent
    from commands.SampleWebRequestEvent import SampleWebRequestOpened
    from commands.SampleCommandEvents import SampleCommandEvent
    from commands.SampleActiveSelectionEvents import SampleActiveSelectionEvent
```

1.5.3 Create the App

To create commands in your addin the first step is to create an instance of *FusionApp*

```
my_addin = apper.FusionApp('ApperSample ', "Autodesk ", False)
```

1.5.4 Standard Commands

Commands are created by subclassing `Fusion360CommandBase` and overriding their *on_xxx* methods.

You add commands to an apper based add-in by calling the *FusionApp.add_command()* function

Sample Command 1

This is adding the command to a panel called "Commands" on the apps Tab in the solid environment.

SampleCommand1 is the basic *Hello World* Fusion 360 command.

It adds a button to the UI that, when clicked, will display a message box with some text.

Command Definition

In the main add-in file we will define the command placement in the UI and define which command the button will be ascociated with. The .. autofunction:: apper.FusionApp.FusionApp.add_command function takes the name of the command, the command class, and a set of options.

```
my_addin.add_command(
    'Sample Command 1',
    SampleCommand1,
    {
        'cmd_description': 'Hello World!',
        'cmd_id': 'sample_cmd_1',
        'workspace': 'FusionSolidEnvironment',
        'toolbar_panel_id': 'Commands',
        'cmd_resources': 'command_icons',
        'command_visible': True,
        'command_promoted': True,
    }
)
```

Learn more about available options by clicking here

Command Class

This command class is defined in a separate file called **SampleCommand1.py**

You can see we are subclassing the Fusion360CommandBase. It is not really important to understand the details of this, but if you just follow this format it will be easy to replicate.

Inside your command class definition you will override one or methods :

- `Fusion360CommandBase.on_create()`
- `Fusion360CommandBase.on_execute()`
- `Fusion360CommandBase.on_preview()`

- `Fusion360CommandBase.on_input_changed()`
- `Fusion360CommandBase.on_destroy()`

In this case we are only overriding the `Fusion360CommandBase.on_execute()` method. So when the user clicks the button the code in this function is immediately executed.

```
import adsk.core
from ..apper import apper
from ..apper.apper import AppObjects


class SampleCommand1(apper.Fusion360CommandBase):
    def on_execute(self, command: adsk.core.Command, inputs: adsk.core.CommandInputs,␣
↪args, input_values):
        ao = AppObjects()
        ao.ui.messageBox("Hello World!")
```

Sample Command 2

Now let's look at a little more complete add-in. In this case we are going to override a number of methods in the `Fusion360CommandBase` class.

on_create

The `Fusion360CommandBase.on_create()` function is executed when the user clicks your icon in the Fusion 360 UI. This is typically where you would define a set of user inputs for your command. The Fusion 360 API makes creating these user interfaces very easy. By getting a reference to the CommandInputs of the command, you can simply add items to the interface. Ass you add items Fusion 360 basically adds them to the bottom of the stack.

```
def on_create(self, command: adsk.core.Command, inputs: adsk.core.CommandInputs):

    # General purpose helper class for quick access to common objects
    ao = AppObjects()

    # Create a default value using a string
    default_value = adsk.core.ValueInput.createByString('1.0 in')

    # Get teh user's current units
    default_units = ao.units_manager.defaultLengthUnits

    # Create a value input.  This will respect units and user defined equation input.
    inputs.addValueInput('value_input_id', '*Sample* Value Input', default_units,␣
↪default_value)

    # Other Input types
    inputs.addBoolValueInput('bool_input_id', '*Sample* Check Box', True)
    inputs.addStringValueInput('string_input_id', '*Sample* String Value', 'Some␣
↪Default Value')
    inputs.addSelectionInput('selection_input_id', '*Sample* Selection', 'Select␣
↪Something')

    # Read Only Text Box
    inputs.addTextBoxCommandInput('text_box_input_id', 'Selection Type: ', 'Nothing␣
↪Selected', 1, True)
```

(continues on next page)

(continued from previous page)

```
    # Create a Drop Down
    drop_down_input = inputs.addDropDownCommandInput('drop_down_input_id', '*Sample*␣
↪Drop Down',
                                                     adsk.core.DropDownStyles.
↪TextListDropDownStyle)
    drop_down_items = drop_down_input.listItems
    drop_down_items.add('List_Item_1', True, '')
    drop_down_items.add('List_Item_2', False, '')
```

on_input_changed

The `Fusion360CommandBase.on_input_changed()` function is executed when the user changes any input value in your ui. This function is typically used to make adjustments to the user interface itself. For example you may want to hide or show certain options based on another input such as a checkbox for "advaced options" or something along those lines. In this case we are updating the text box text with the object type of whatever the user has selected. Note code in this method will not affect the graphics window. If you want to update the displayed geometry you should use the `Fusion360CommandBase.on_preview()` method.

1.6 Events

CHAPTER

TWO

THE API DOCUMENTATION / GUIDE

If you are looking for information on a specific function, class, or event, this part of the documentation is for you.

2.1 Developer Interface

This part of the documentation covers all the interfaces of Apper.

2.1.1 Core Apper Modules

The core Apper functionality can be accessed by sub-classing these 3 classes. Step one is to create an instance of the *FusionApp* object. Step two is to add instances of `apper.Fusion360CommandBase` and `apper.PaletteCommandBase` classes. Each instance of these classes will be a new command in your add-in.

class apper.**FusionApp**(*name*, *company*, *debug*)

Base class for creating a Fusion 360 Add-in

Parameters

- **name** (str) – The name of the addin
- **company** (str) – the name of your company or organization
- **debug** (bool) – set this flag as True to enable more interactive feedback when developing.

add_command(*name*, *command_class*, *options*)

Adds a command to the application

Parameters

- **name** (str) – The name of the command
- **command_class** (Any) – This should be your subclass of apper.Fusion360CommandBase or apper.PaletteCommandBase
- **options** (dict) – Set of options for the command see the full set of options

add_command_event(*event_id*, *event_type*, *event_class*)

Register a workspace event that can respond to various workspace actions

Parameters

- **event_id** (str) – A unique identifier for the event
- **event_type** (Any) – One of [UserInterface.commandCreated, UserInterface.commandStarting, UserInterface.commandTerminated]
- **event_class** (Any) – Your subclass of apper.Fusion360CommandEvent class

add_custom_event(*event_id*, *event_class*, *auto_start=True*)
Register a custom event to respond to a function running in a new thread

Parameters

- **event_id** (str) – A unique identifier for the event
- **event_class** (Any) – Your subclass of apper.Fusion360CustomThread
- **auto_start** (bool) – Whether the thread should start when the addin starts

add_custom_event_no_thread(*event_id*, *event_class*)
Register a custom event

Parameters

- **event_id** (str) – A unique identifier for the event
- **event_class** (Any) – Your subclass of apper.Fusion360CustomThread

add_custom_feature(*name*, *feature_class*, *options*)
Register a workspace event that can respond to various workspace actions

Parameters

- **name** (str) – The name of the command
- **feature_class** (Any) – This should be your subclass of apper.Fusion360CustomFeatureBase
- **options** (dict) – Set of options for the command see the full set of options

Return type Any

add_document_event(*event_id*, *event_type*, *event_class*)
Register a document event that can respond to various document actions

Parameters

- **event_id** (str) – A unique identifier for the event
- **event_type** (DocumentEvent) – Any document event in the current application
- **event_class** (Any) – Your subclass of apper.Fusion360DocumentEvent

add_web_request_event(*event_id*, *event_type*, *event_class*)
Register a workspace event that can respond to various workspace actions

Parameters

- **event_id** (str) – A unique identifier for the event
- **event_class** (Any) – Your subclass of apper.Fusion360WebRequestEvent
- **event_type** (WebRequestEvent) – Opened or Inserting from URL event type such as (app.openedFromURL)

add_workspace_event(*event_id*, *workspace_name*, *event_class*)
Register a workspace event that can respond to various workspace actions

Parameters

- **event_id** (str) – A unique identifier for the event
- **workspace_name** (str) – name of the workspace (i.e.
- **event_class** (Any) – Your subclass of apper.Fusion360WorkspaceEvent

check_for_updates()
Not Implemented

command_id_from_name(*name*)
Returns the full cmd_id defined by apper

Parameters **name** (str) – this is the value set in options for cmd_id

Return type Optional[str]

Returns The full cmd_id (i.e. CompanyName_AppName_cmd_id)

get_all_preferences()
Gets all preferences stored for this application

Return type dict

Returns All preferences as a dictionary

get_group_preferences(*group_name*)
Gets preferences for a particular group (typically a given command)

Parameters **group_name** (str) – name of parent group in which to store preferences

Return type dict

Returns A dictionary of just the options associated to this particular group

initialize_preferences(*defaults*, *force=False*)
Initializes preferences for the application

Parameters

- **defaults** (dict) – a default set of preferences
- **force** – If True, any existing user preferences will be over-written

Returns "Created", "Exists", or "Failed"

Return type A string with possible values

static read_json_file(*file_name*)
Static method to read a json file and return a dictionary object

Will fail if the input file cannot be interpreted as a JSON object

Returns Input file as a dictionary

run_app()
Runs the Addin

save_preferences(*group_name*, *new_group_preferences*, *merge*)
Saves preferences for the application

Parameters

- **group_name** (str) – name of parent group in which to store preferences
- **new_group_preferences** (dict) – Dictionary of preferences to save
- **merge** (bool) – If True then the new preferences in the group will be merged, if False all old values are deleted

Returns "Updated", "Created", or "Failed"

Return type A string with possible values

stop_app()
Stops the Addin and cleans up all of the created UI elements

2.1.2 Other Modules

Fusion360Utilities.py

Tools to leverage when creating a Fusion 360 Add-in

copyright

(c) 2019 by Patrick Rainsberry.

license Apache 2.0, see LICENSE for more details.

class apper.Fusion360Utilities.**AppObjects**
The AppObjects class wraps many common application objects required when writing a Fusion 360 Addin.

property cam
adsk.cam.CAM from the active document

Note if the document has never been activated in the CAM environment this will return None

Returns: adsk.cam.CAM from the active document

Return type Optional[CAM]

property design
adsk.fusion.Design from the active document

Returns: adsk.fusion.Design from the active document

Return type Optional[Design]

property document
adsk.fusion.Design from the active document

Returns: adsk.fusion.Design from the active document

Return type Optional[Document]

property export_manager
adsk.fusion.ExportManager from the active document

Returns: adsk.fusion.ExportManager from the active document

Return type Optional[ExportManager]

property f_units_manager
adsk.fusion.FusionUnitsManager from the active document.

Only work in design environment.

Returns: adsk.fusion.FusionUnitsManager or None if in a different workspace than design.

Return type Optional[FusionUnitsManager]

property product
adsk.fusion.Design from the active document

Returns: adsk.fusion.Design from the active document

Return type Optional[Product]

property root_comp
Every adsk.fusion.Design has exactly one Root Component

It should also be noted that the Root Component in the Design does not have an associated Occurrence

Returns: The Root Component of the adsk.fusion.Design

Return type `Optional[Component]`

property time_line
adsk.fusion.Timeline from the active adsk.fusion.Design

Returns: adsk.fusion.Timeline from the active adsk.fusion.Design

Return type `Optional[Timeline]`

property units_manager
adsk.core.UnitsManager from the active document

If not in an active document with design workspace active, will return adsk.core.UnitsManager if possible

Returns: adsk.fusion.FusionUnitsManager or adsk.core.UnitsManager if in a different workspace than design.

Return type `Optional[UnitsManager]`

`apper.Fusion360Utilities.`**combine_feature**(*target_body*, *tool_bodies*, *operation*)
Creates Combine Feature in target with all tool bodies as source

Parameters

- **target_body** (`BRepBody`) – Target body for the combine feature
- **tool_bodies** (`List[BRepBody]`) – A list of tool bodies for the combine
- **operation** (`FeatureOperations`) – An Enumerator defining the feature operation type

`apper.Fusion360Utilities.`**create_component**(*target_component*, *name*)
Creates a new empty component in the target component

Parameters

- **target_component** (`Component`) – The target component for the new component
- **name** (`str`) – The name of the new component

Return type `Occurrence`

Returns The reference to the occurrence of the newly created component.

`apper.Fusion360Utilities.`**end_group**(*start_index*)
Ends a adsk.fusion.TimelineGroup

start_index: adsk.fusion.TimelineGroup index that is returned from start_group

`apper.Fusion360Utilities.`**extrude_all_profiles**(*sketch*, *distance*, *component*, *operation*)
Create extrude features of all profiles in a sketch

The new feature will be created in the given target component and extruded by a distance

Parameters

- **sketch** (`Sketch`) – The sketch from which to get profiles
- **distance** (`float`) – The distance to extrude the profiles.
- **component** (`Component`) – The target component for the extrude feature

- **operation** (FeatureOperations) – The feature operation type from enumerator.

Return type ExtrudeFeature

Returns The new extrude feature.

apper.Fusion360Utilities.**get_a_uuid**()
Gets a base 64 uuid

Return type str

Returns The id that was generated

apper.Fusion360Utilities.**get_default_dir**(*app_name*)
Creates a directory in the user's home folder to store data related to this app

Parameters **app_name** (*str*) – Name of the Application

apper.Fusion360Utilities.**get_item_by_id**(*this_item_id*, *app_name*)
Returns an item based on the assigned ID set with *item_id*

Parameters

- **this_item_id** (str) – The unique id generated originally by calling *item_id*
- **app_name** (str) – Name of the Application

Return type Base

Returns The Fusion 360 object that the id attribute was attached to.

apper.Fusion360Utilities.**get_log_file**(*app_name*)
Gets the filename for a default log file

Parameters **app_name** (str) – Name of the Application

apper.Fusion360Utilities.**get_log_file_name**(*app_name*)
Gets the filename for a default log file

Parameters **app_name** (str) – Name of the Application

apper.Fusion360Utilities.**get_settings_file**(*app_name*)
Create (or get) a settings file name in the default app directory

Parameters **app_name** (str) – Name of the Application

apper.Fusion360Utilities.**get_std_err_file**(*app_name*)
Get temporary stderr file for the app

Parameters **app_name** (str) – Name of the Application

apper.Fusion360Utilities.**get_std_out_file**(*app_name*)
Get temporary stdout file for the app

Parameters **app_name** (str) – Name of the Application

apper.Fusion360Utilities.**import_dxf**(*dxf_file*, *component*, *plane*, *is_single_sketch_result=False*)
Import dxf file with one sketch per layer.

Parameters

- **dxf_file** (str) – The full path to the dxf file
- **component** (Component) – The target component for the new sketch(es)
- **plane** (Union[ConstructionPlane, BRepFace]) – The plane on which to import the DXF file.

- **plane** – The plane on which to import the DXF file.
- **is_single_sketch_result** (bool) – If true will collapse all dxf layers to a single sketch.

Return type ObjectCollection

Returns An ObjectCollection of the created sketches

apper.Fusion360Utilities.**item_id**(*item*, *group_name*)
Gets (and possibly assigns) a unique identifier (UUID) to any item in Fusion 360

Parameters

- **item** (Base) – Any Fusion Object that supports attributes
- **group_name** (str) – Name of the Attribute Group (typically use app_name)

Return type str

Returns The id that was generated or was previously existing

class apper.Fusion360Utilities.**lib_import**(*library_folder*)
The lib_import class is a wrapper class to allow temporary import of a local library directory

By default it assumes there is a folder named 'lib' in the add-in root directory.

First install a 3rd party library (such as requests) to this directory.

```
# Assuming you are in the add-in root directory (sudo may not be required...)
sudo python3 -m pip install -t ./lib requests
```

Then you can temporarily import the library before making a call to the requests function. To do this use the *@apper.lib_import(...)* decorator on a function that uses the library.

Here is an example function for using Requests:

```
@apper.lib_import(config.app_path)
def make_request(url, headers):
    import requests
    r = requests.get(url, headers=headers)
    return r
```

Parameters

- **app_path** (*str*) – The root path of the addin. Should be dynamically calculated.
- **library_folder** (str, optional) – Library folder name (relative to app root). Defaults to 'lib'

apper.Fusion360Utilities.**open_doc**(*data_file*)
Simple wrapper to open a dataFile in the application window

Parameters **data_file** (DataFile) – The data file to open

apper.Fusion360Utilities.**read_settings**(*app_name*)
Read a settings file into the default directory for the app

Parameters **app_name** (str) – Name of the Application

apper.Fusion360Utilities.**rect_body_pattern**(*target_component*, *bodies*, *x_axis*, *y_axis*, *x_qty*, *x_distance*, *y_qty*, *y_distance*)
Creates rectangle pattern of bodies based on vectors

Parameters

- **target_component** (Component) – Component in which to create the patern
- **bodies** (List[BRepBody]) – bodies to pattern
- **x_axis** (Vector3D) – vector defining direction 1
- **y_axis** (Vector3D) – vector defining direction 2
- **x_qty** (int) – Number of instances in direction 1
- **x_distance** (float) – Distance between instances in direction 1
- **y_qty** (int) – Number of instances in direction 2
- **y_distance** (float) – Distance between instances in direction 2

Return type ObjectCollection

apper.Fusion360Utilities.**remove_item_id**(*item*, *group_name*)
Gets (and possibly assigns) a unique identifier (UUID) to any item in Fusion 360

Parameters

- **item** (Base) – Any Fusion Object that supports attributes
- **group_name** (str) – Name of the Attribute Group (typically use app_name)

Return type bool

Returns True if successful and False if it failed

apper.Fusion360Utilities.**sketch_by_name**(*sketches*, *name*)
Finds a sketch by name in a list of sketches

Useful for parsing a collection of sketches such as DXF import results.

Parameters

- **sketches** (Sketches) – A list of sketches. (Likely would be all sketches in active document).
- **name** (str) – The name of the sketch to find.

Return type Sketch

Returns The sketch matching the name if it is found.

apper.Fusion360Utilities.**start_group**()
Starts a time line group

Return type int

Returns The index of the adsk.fusion.Timeline where the adsk.fusion.TimelineGroup will begin

apper.Fusion360Utilities.**write_settings**(*app_name*, *settings*)
Write a settings file into the default directory for the app

Parameters

- **app_name** (str) – Name of the Application
- **settings** (dict) – Stores a dictionary as a json string

Fusion360DebugUtilities.py

Utilities to aid in debugging a Fusion 360 Addin

copyright

(c) 2019 by Patrick Rainsberry.

license Apache 2.0, see LICENSE for more details.

apper.Fusion360DebugUtilities.**get_log_file_name**()
Creates directory and returns file name for log file :param log: tbd

apper.Fusion360DebugUtilities.**perf_log**(*log*, *function_reference*, *command*, *identifier=''*)
Performance time logging function :param log: :param function_reference: :param command: :param identifier:

apper.Fusion360DebugUtilities.**perf_message**(*log*)
Performance time logging function :param log: tbd

apper.Fusion360DebugUtilities.**variable_message**(*variable*, *extra_info=''*)
Displays the value of any single variable as long as the value can be converted to text

Parameters

- **variable** – variable to print
- **extra_info** – Any other info to display in the message box

apper.Fusion360DebugUtilities.**variables_message**(*variables*)
Print a list of list of variables

Format of variables should be [[Variable name 1, variable value 1], [Variable name 2, variable value 2], . . .]

Parameters **variables** (list) – A list of lists of any string based variables from your add-in.

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

a

apper, 13
apper.Fusion360DebugUtilities, 20
apper.Fusion360Utilities, 16

INDEX

A

add_command() (*apper.FusionApp method*), 13
add_command_event() (*apper.FusionApp method*), 13
add_custom_event() (*apper.FusionApp method*), 13
add_custom_event_no_thread() (*apper.FusionApp method*), 14
add_custom_feature() (*apper.FusionApp method*), 14
add_document_event() (*apper.FusionApp method*), 14
add_web_request_event() (*apper.FusionApp method*), 14
add_workspace_event() (*apper.FusionApp method*), 14
apper
    module, 13
apper.Fusion360DebugUtilities
    module, 20
apper.Fusion360Utilities
    module, 16
AppObjects (*class in apper.Fusion360Utilities*), 16

C

cam() (*apper.Fusion360Utilities.AppObjects property*), 16
check_for_updates() (*apper.FusionApp method*), 14
combine_feature() (*in module apper.Fusion360Utilities*), 17
command_id_from_name() (*apper.FusionApp method*), 15
create_component() (*in module apper.Fusion360Utilities*), 17

D

design() (*apper.Fusion360Utilities.AppObjects property*), 16
document() (*apper.Fusion360Utilities.AppObjects property*), 16

E

end_group() (*in module apper.Fusion360Utilities*), 17
export_manager() (*apper.Fusion360Utilities.AppObjects property*), 16
extrude_all_profiles() (*in module apper.Fusion360Utilities*), 17

F

f_units_manager() (*apper.Fusion360Utilities.AppObjects property*), 16
FusionApp (*class in apper*), 13

G

get_a_uuid() (*in module apper.Fusion360Utilities*), 18
get_all_preferences() (*apper.FusionApp method*), 15
get_default_dir() (*in module apper.Fusion360Utilities*), 18
get_group_preferences() (*apper.FusionApp method*), 15
get_item_by_id() (*in module apper.Fusion360Utilities*), 18
get_log_file() (*in module apper.Fusion360Utilities*), 18
get_log_file_name() (*in module apper.Fusion360DebugUtilities*), 21
get_log_file_name() (*in module apper.Fusion360Utilities*), 18
get_settings_file() (*in module apper.Fusion360Utilities*), 18
get_std_err_file() (*in module apper.Fusion360Utilities*), 18
get_std_out_file() (*in module apper.Fusion360Utilities*), 18

I

import_dxf() (*in module apper.Fusion360Utilities*), 18

initialize_preferences() (*apper.FusionApp method*), 15
item_id() (*in module apper.Fusion360Utilities*), 19

L

lib_import (*class in apper.Fusion360Utilities*), 19

M

module
    apper, 13
    apper.Fusion360DebugUtilities, 20
    apper.Fusion360Utilities, 16

O

open_doc() (*in module apper.Fusion360Utilities*), 19

P

perf_log() (*in module apper.Fusion360DebugUtilities*), 21
perf_message() (*in module apper.Fusion360DebugUtilities*), 21
product() (*apper.Fusion360Utilities.AppObjects property*), 16

R

read_json_file() (*apper.FusionApp static method*), 15
read_settings() (*in module apper.Fusion360Utilities*), 19
rect_body_pattern() (*in module apper.Fusion360Utilities*), 19
remove_item_id() (*in module apper.Fusion360Utilities*), 20
root_comp() (*apper.Fusion360Utilities.AppObjects property*), 16
run_app() (*apper.FusionApp method*), 15

S

save_preferences() (*apper.FusionApp method*), 15
sketch_by_name() (*in module apper.Fusion360Utilities*), 20
start_group() (*in module apper.Fusion360Utilities*), 20
stop_app() (*apper.FusionApp method*), 15

T

time_line() (*apper.Fusion360Utilities.AppObjects property*), 17

U

units_manager() (*apper.Fusion360Utilities.AppObjects property*), 17

V

variable_message() (*in module apper.Fusion360DebugUtilities*), 21
variables_message() (*in module apper.Fusion360DebugUtilities*), 21

W

write_settings() (*in module apper.Fusion360Utilities*), 20