Advances in Artificial Intelligence, Big Data and Algorithms
G. Grigoras and P. Lorenz (Eds.)

doi:10.3233/FAIA230799

Research on Application Automation Operations Based on Win32com

Jing LI[1], Peizhang WANG, Qian LI, Yongle HE, Lin SUN, Yi SUN, Pinwang ZHAO and Lu JIA
Huaishuling 4#, Network Information Department, China North Vehicle Research Institute; Fengtai, Beijing, China

Abstract. This article aims to explore the method of utilizing the win32com module to interact with application API for achieving automation operations. Taking the AutoCAD application as an example, it introduces the automation operations of common tasks (such as file opening, property retrieval, etc.) by invoking the API interface. By studying the API documentation of AutoCAD and combining it with the win32com. client. Dispatch function, sample code is written to demonstrate the application of this technique in automation operations.

Keywords. win32com, python, pywin32, Automation operations, AutoCAD

1. Introduction

With the widespread use of computer applications, achieving automation operations for applications has become an important means to improve work efficiency and reduce labor costs. The win32com module, as a powerful tool, enables interaction with the COM interface of applications to achieve the goal of automation operations. Win32com can only be used on the Windows operating system. This paper takes the AutoCAD application as an example and explores how to interact with its API using the win32com.client.Dispatch function to achieve automation operations.

2. Brief Introduction, Installation, Usage Steps, and Applications of win32com

2.1. What is win32com

The win32com module is a module in the Python language that provides automation and interaction with COM (Component Object Model) components on the Windows platform. COM is an object-oriented binary interface standard used to achieve interoperability between different applications[1]. Every software installed on Windows usually registers its important components with the system[2]. Built on the Python extension library pywin32, the win32com module enables access to the Windows API and COM interfaces, allowing Python programs to interact with various COM-supported applications. With

[1]Corresponding Author: Jing LI, Associate Researcher, China North Vehicle Research Institute; e-mail: 30952197@qq.com

win32com, it is possible to create, manipulate, and control COM objects, invoke methods, and access properties, implement automated operations.

It should be noted that the win32com module depends on the third-party module pywin32. The pywin32 can be installed by executing "pip install pywin32" at Command Prompt. To verify the installation, you can enter "import win32api" fand if there are no errors occur, it indicates the installation has been successful. But if the error "ImportError: DLL load failed while importing win32api: The specified module could not be found." occurs, you will need to execute "pip install pypiwin32", and then find the full path of the file "pywin32_postinstall.py", which is generally under the Scripts folder of the installed directory, and execute "python pywin32_postinstall.py install" under that path to install successfully. If you are using the Anaconda Python environment, there is no need for additional installation of third-party libraries as pywin32 is already included in Anaconda.

2.2. How to use win32com

After setting up the environment, using the win32com module requires these following steps:

- Create a COM object:

Use the win32com.client.Dispatch function to create a COM object associated with a specific application. This function takes the ProgID (Programmatic Identifier) or CLSID (Class Identifier) of the application as a parameter to instantiate the corresponding COM object[3].

Common Applications and the sentence which create the COM objects by python is shown in table 1.

Table 1. Examples of common applications and how to create the COM objects by python

Application	Create the COM Objects by Python
Doc	doc= win32com.client.Dispatch("Word.Application")
Excel	xl= win32com.client.Dispatch("Excel.Application")
PowerPoint	ppt= win32com.client.Dispatch("PowerPoint.Application")
AutoCAD	acad = win32com.client.Dispatch("AutoCAD.Application")
Internet Explorer	ie = win32com.client.Dispatch("InternetExplorer.Application")

For example, when passing "AutoCAD.Application" as a parameter, win32com.client. Dispatch will look up the registry to obtain the ProgID of the COM component associated with the AutoCAD application. ProgID is a human readable string name used to identify COM components.

- Operating COM objects:

After creating a COM object, you can invoke the methods and access the properties of the application. For example, you can use the methods of the COM object to open files, modify properties and perform other operations. To understand the automation interfaces and methods provided by the application, you can refer to its API documentation or developer resources. These documents serve as the foundation for performing automation operations.

However, not all applications provide automation interfaces. Before using win32com.client.Dispatch, it is important to determine whether the desired application supports COM automation. This can be done by consulting the API documentation or other developer resources to understand its automation capabilities and interfaces. Additionally, when interacting with an application for automation, it is necessary to know and comply with the automation rules and security restrictions imposed by the application. Some applications may require security settings or specific authorization to allow automation access by external programs.

- Event handling:

In general, COM components support an event-driven programming model. You can use the "WithEvents" method to register event handlers and perform corresponding actions when events are triggered by the COM component.

- Resource release:

After completing the operations, make sure to close the COM objects to free resources. You can use the Close method of the COM object to perform the closing or releasing, to avoid memory leaks and unstable behavior.

2.3. The example program: Classifying DWG Drawings by Sheet Size Using win32com

During the engineering design process, Numerous CAD drawings are generated to record the structural information and parameter details of the products. These historical drawings are mainly in AutoCAD's DWG format, which doesn't directly display the width and height of the drawings. Designers need to manually open each file and place the mouse in the top right and bottom left corners of the drawing to confirm the sheet size by subtracting these two coordinates. For example, if it is calculated to be (420, 297), it can be confirmed as an A3 sheet size. Due to the enormous quantity of historical drawings, manually checking the sheet size information one by one is time-consuming and error-prone. To reduce the workload on engineers, improve work efficiency, and enhance accuracy, it is necessary to develop a program to extract the width and height information of DWG format drawings in batch. In this context, I wrote an example program based on win32com using Python script language[4].

After querying the API documentation provided by AUTODESK, decisive system variables related to the DWG drawings sheet size and their description were found, as shown in table 2:

Table 2. System variables and their description

System Variable	Description
EXTMIN	Stores the lower-left point of the drawing extents.
EXTMAX	Stores the upper-right point of the drawing extents.

The Python statement that calls system variables is:

```
acad = win32com.client.Dispatch("AutoCAD.Application")
cad = acad.Documents.Open(file)
max = cad.GetVariable("EXTMAX")
min=cad.GetVariable("EXTMIN")
```

It is worth mentioning that AUTODESK provides "LIMITS" command with the function of "Sets an invisible rectangular boundary in the drawing area that can limit the grid display and limit clicking or entering point locations[5]". However, the default layout size for new DWG drawings is (420, 297). If the actual layout size is larger than this size, the designer will modify it to the actual size on their own; But if it is smaller than this size, such as (210, 297), the designer may skip it without making any modifications, which may result in incorrect size read by "cad.LIMITS". Therefore, it is more accurate to use the difference between the system variables "EXTMAX" and "EXTMIN" to confirm the DWG drawings sheet size.

The flowchart of the example program is shown in figure 1 as follows:

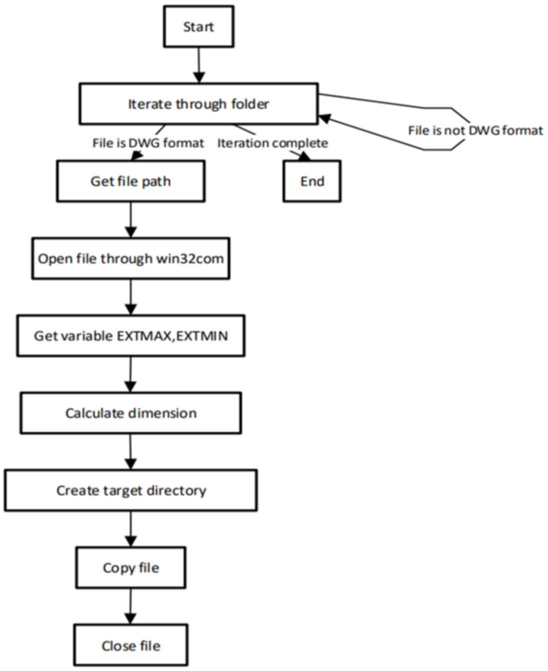
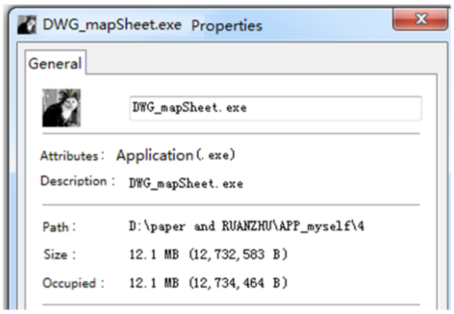


Figure 1. The flowchart of the example program

The written Python script is converted into an executable (.exe) file using the pyinstaller library, It supports Python 3.7 and newer, bundling a Python application and all its dependencies into a single package[6], enabling it to run directly on the Windows 7 operating system. The executable file can effortlessly be delivered and provided by the developer to others, empowering them to make utilize of the statistical functionalities of the program without installing a Python interpreter or any added installation steps. Users can simply double click the executable file to perform the statistical and classification work. This greatly enhances the usability and accessibility of the tool. As indicated in figure 2, the size of this executable file is only 12.1 MB.


DWG_mapSheet.exe Properties

General

DWG_mapSheet. exe

Attributes : Application (. exe)

Description : DWG_mapSheet. exe

Path : D:\paper and RUANZHU\APP_myself\4

Size : 12. 1 MB (12, 732, 583 B)

Occupied : 12. 1 MB (12, 734, 464 B)

Figure 2. The size of the tool after packaged into an EXE

3. Experimental Results and Discussion

The effectiveness of interacting with AutoCAD's API using the win32com module was validated through experiments and testing. We developed a sample program demonstrating how to automate operations using the win32com module with AutoCAD's API. This sample program traverses DWG files in a specified directory. It automatically opens each file, retrieves its properties, calculates the sheet size, closes the file, and then copies it to the corresponding folder named after the sheet size. The program runtime for handling different drawing sizes and page numbers is shown in the Table 3 below.

Table 3. Statistical table of program Runtime and number of drawings

Number of all	A0	A1	A2	A3	A4	Non-standard	Time(seconds)
32	3	3	3	10	12	1	17.002803
50	3	3	5	17	20	2	26.875725
91	4	4	10	32	36	5	53.364305
131	6	5	11	54	50	5	88.788696

The following observations were made during the experiments:

- Successful creation of a COM object associated with the AutoCAD application using the win32com.client.Dispatch function, establishing a connection with AutoCAD.
- Implementation of automatic file opening, extraction of drawing width and height properties, and other functionalities by calling AutoCAD's API methods and properties such as "acad.Documents.Open()", "cad.GetVariable()" and "cad.Close()".
- Development of code for looping through the file list, enabling automation of processing for all DWG files that meet the specified conditions in the selected directory.
- The automation process exhibited swift and accurate operations, resulting in significant savings in manual operation time and labor costs.

4. Conclusions

This article's example program showcases the application of automating operations by using win32com module and the API of AutoCAD. The win32com module is one of the essential tools for implementing automation operations and integrating with other applications, providing developers with a wide range of applications and creative solutions. By understanding the API documentation of the target application, we can call appropriate methods and properties to achieve various automated tasks. This method is particularly useful for batch processing, large-scale data processing, and repetitive tasks. Automate tedious tasks, liberate productivity, and improve efficiency. Taking the development of an automated tool for extracting sheet size information from DWG drawings as an example, it addresses the issue of inefficient processing of large-scale drawing data. The tool enables automated extraction and processing of DWG drawings, swiftly retrieving layout width and height information in bulk, and generating output results. This provides valuable data support for drawing management and categorization, enhancing data processing efficiency and accuracy.

Future research directions include exploring automation operations in other applications in conjunction with the win32com, different applications may offer various automation interfaces and functionalities, Additionally, we can optimize the code of automation operations to enhance execution efficiency and stability, as well as exploring the integration of machine learning with automation operations to achieve more intelligent automation. For applications without COM interfaces or with restrictions, alternative solutions or technologies need to be considered.

In summary, by researching and utilizing the win32com module and the application's API for automation, it is possible to achieve efficient and accurate batch task processing. This research provides a reference for similar automation tasks, has a broad application prospect and promotion application value, and plays a positive role in promoting the development of big data and artificial intelligence and other technologies in practical applications.

References

[1] Chuanlu Huang, Zhiqiang Zhang, *Zero Basis Python Second Edition*, China Machine Press, Beijing, 2020.
[2] Preston Miller, Chapin Bryce, *Learning Python for Forensics Second Edition*, Packet Publishing, Birmingham, 2019.
[3] Luciano Ramalho, *Fluent Python*, O'Reilly Media, California, 2015.
[4] HUANG Huan, XU Tan, Development of Data Automatic Statistics for Flight Simulation Training Equipment, *Computer Knowledge and Technology* **21**(2020), 203-205.
[5] AUTODESK, https://help.autodesk.com/view/OARX/2019/ENU/, 2019.
[6] David C, PyInstaller Manual, https://pyinstaller.org/en/stable/,2023.