Advanced Scripting   
Interact with COM using PowerShell

Last Updated: 4/13/2024 12:46 PM Version 1  
Document Prepared for: CYBER360 Student

# Name Click here to enter name ID Click here to enter id

# Instructions

Answer all questions directly in this document. You will save and upload this completed document as your homework submission.

# Overview

Microsoft introduced Component Object Model (COM) as an object-based framework for different applications to interact with each other.

# Requirements

Microsoft Windows.

# Setup

Launch Windows PowerShell (either Core or Desktop).

# Task 1 — Automate desktop operations

You can use PowerShell commands to automate the Windows desktop interface.

## Steps

1. At your PowerShell prompt, create an object that lets you interact with the component object model (COM) of the native graphical user interface in Microsoft Windows. Enter:   
   **$MyDesktop = New-Object -ComObject Shell.Application**
2. What is the actual class of this PowerShell object? Enter:   
   **$MyDesktop.GetType().Name**   
   Your result: Click or tap here to enter text.
3. Discover the methods and properties of this COM object. Enter:   
   **$MyDesktop | Get-Member**
4. Let’s try out some of the methods. To minimize all open windows, enter:   
   **$MyDesktop.MinimizeAll()**
5. Bring back PowerShell from its minimized representation on the taskbar. Then enter:   
   **$MyDesktop.UndoMinimizeALL()**   
   That should restore all of the other windows that were minimized.
6. Enter:   
   **$MyDesktop.SearchCommand()**   
   This should activate the search interface of the Windows GUI.
   1. Type the first few characters of File Explorer until **File Explorer** appears under **Best match**, then press the [Enter] key to launch it.
   2. Close the File Explorer window.

# Task 2 — Use PowerShell to find and explore an application with COM

## Steps

1. Every “*programmatic identifier*” (ProgID) for a COM object has an entry in the Windows Registry. To list them all, enter the following (all on one line):   
   **Get-ChildItem -Path 'HKLM:\Software\Classes\' -ErrorAction SilentlyContinue | Select-Object PSChildName**
   1. Most of them are merely components, not full applications that can be automated. Insert a filter in the previous pipeline that only passes ProgIDs that end with **.Application**:   
      **Get-ChildItem -Path 'HKLM:\Software\Classes\' -ErrorAction SilentlyContinue | ? { $\_.PSChildName -match '^\w+\.Application$' } | Select-Object PSChildName**
   2. It’s worth your time to carefully study the **Where-Object** cmdlet we inserted into the pipeline.   
       - Remember that **?** is a convenient alias for **Where-Object**.  
       - Remember **$\_** is the “loop variable” that contains each item input through the pipeline.   
       - (In this case, **$\_** contains each Registry node under   
       **HKEY\_LOCAL\_MACHINE\Software\Classes**, one at a time.)  
       - Remember that the **-match** operator compares a string against a regular expression.  
       - The regular expression used here is **^\w+\.Application$**. Let’s analyze it:   
       **^** matches the beginning of the string.   
       **\w+** matches one or more words (sequences of alphanumeric characters).   
       **\.** matches the “dot” (**.**, period) character.  
       (Don’t just use **.** because in a regex the dot matches *any* character. Use **\.**.)  
       **$** matches the end of the string.   
       - So that’s how this regular expression matches ProgIDs that end with **.Application**.
2. Look through your filtered output. Find a ProgID of an Application that looks interesting, and create a new COM object of that class. For example, one of the results on my computer was **PowerPoint.Application**, so I thought it might be nifty to try that class:   
   **$nifty = New-Object -ComObject PowerPoint.Application**   
   What **New-Object -ComObject** class did you try? Click or tap here to enter text. (If your chosen ProgID fails or produces an exception, try a different one.)
3. Discover the methods and properties of your selected class, by sending your new COM object through a pipe into the **Get-Member** cmdlet:   
   **$nifty | Get-Member**   
   List two members (that’s two methods, or two properties, or one of each) that you can access in your COM object: Click or tap here to enter text.
4. Take a moment to try accessing some of the members of your COM object, and see if you can figure out how to use them.
   * 1. What is one of the commands you tried? Click or tap here to enter text.
     2. What was the effect, output, or exception produced by that command? Click or tap here to enter text.
5. *IMPORTANT:* check whether your COM object has a method named **Quit()**. If it does, then when you created your object, it probably actually launched a software program, which is now invisibly running and consuming resources on your computer! Before finishing this task, be sure to use the **Quit()** method to close that application. For example, to close the PowerPoint instance that was launched when I created my PowerPoint.Application COM object, I entered:   
   **$nifty.Quit()**   
   and after that   
   **$nifty = $null**

# Task 3 (optional) — Use COM to automate Excel tasks

(If your computer runs macOS or Linux, or if you cannot install Microsoft Office on your computer, skip this task.)

## Steps

1. Launch the Task Manager.
2. At your PowerShell prompt, create a COM object to interact with Microsoft Excel:   
   **$e = New-Object -ComObject Excel.Application**
3. Create a new Excel workbook:   
   **$MyWorkbook = $e.Workbooks.Add()**
4. Access the spreadsheet in the new workbook:   
   **$MySheet = $MyWorkBook.Worksheets.Item(1)**
5. Give the spreadsheet a name:   
   **$MySheet.name = 'Howdy'**
6. Put content in the top left cell of the spreadsheet:   
   **$MySheet.Cells.Item(1,1) = 'Howdy there!'**
7. Save the workbook to a temporary file:   
   **$MyWorkbook.SaveAs('C:\Temp\Howdy.xlsx')**   
   (If you want it saved to some other location, feel free to adjust the path string.)
8. Close the Excel processes that were launched in step 1:   
   **$e.Quit()**
9. Use the start command to launch Excel again, to visibly see your results:   
   **start C:\Temp\Howdy.xlsx**   
   Close Excel when you’re satisfied.

# Deliverable

Upload this document with completed answers to I-Learn Canvas.