***Лабораторна робота №14***

***«Робота з COM- та .NET-об’єктами у сценаріях PowerShell»***

***(Термін виконання – 2 тижні!)***

***Робота з СОМ-об'єктами***

Працюючи в **PowerShell**, можна ідентифікувати COM-об'єкти за їх програмними ідентифікаторами (**ProgID**) - символьними псевдонімами, що призначаються при реєстрації об'єктів в системі. Згідно із загальноприйнятої угоди ідентифікатори **ProgID** мають такий вигляд: ***Бібліотека\_типів.Клас.Версія*** або просто ***Бібліотека\_типів.Клас***

Достатньо часто ім'я бібліотеки типів збігається з ім'ям програми, що є сервером COM-об'єктів. Після точки в ProgID вказується ім'я класу, що містить властивості і методи COM-об'єкта, доступні для використання іншими додатками. Номер версії при створенні екземплярів СОМ-об'єктів, як правило, не використовується. Ось кілька прикладів **ProgID**: ***IntenetExplorer.Application*** (додаток **Internet Explorer**). ***Word.Application*** (додаток **Microsoft Word**), ***WScript.Shell*** (клас **Shell** з об'єктної моделі сервера сценаріїв **Windows Script Host**).

У **PowerShell** є командлет ***New-Object***, що дозволяє, зокрема, створювати екземпляри зовнішніх COM-об'єктів, вказуючи відповідний **ProgID** як значення параметра ***-ComObject***. Наприклад, екземпляр СОМ-об'єкту з програмним ідентифікатором ***WScript.Shell*** створюється наступним чином:

***PS С:\> $Shell = New-Object -ComObject WScript.Shell***

Подивимося, які властивості і методи є у СОМ-об'єкту ***WScript.Shell***. Для цього скористаємося, як зазвичай, командлетом ***Get-Member***, передавши йому по конвеєру змінну ***$Shell***, в якій збережене посилання на даний СОМ-об'єкт:

***PS C:\Users\Глеб> $Shell | Get-Member***

***TypeName: System.\_\_ComObject#{41904400-be18-11d3-a28b-00104bd35090}***

***Name MemberType Definition***

***---- ---------- ----------***

***AppActivate Method bool AppActivate (Variant, Variant)***

***CreateShortcut Method IDispatch CreateShortcut (string)***

***Exec Method IWshExec Exec (string)***

***-------------------------------------------------------------------------***

***Об'єктні моделі Microsoft Word і Excel***

Одними з найпоширеніших і часто використовуваних серверів автоматизації в Windows є додатки пакета Microsoft Office. Ми розглянемо на прикладах, яким чином можна виводити з **PowerShell** інформацію в дві найбільш поширені програми цього пакета: **Microsoft Word** і **Microsoft Excel**.

Хоча об'єктні моделі додатків Microsoft Office досить складні, вони схожі один на одного, причому для практичних цілей достатньо зрозуміти принцип роботи з декількома ключовими об'єктами.

На самому верхньому рівні об'єктної моделі **Word** знаходиться об'єкт ***Application***, який представляє безпосередньо сам додаток **Word** і містить (як властивості) всі інші об'єкти. Таким чином, об’єкт ***Application*** використовується для отримання доступу до будь-якого іншого об’єкту **Word**.

Сімейство ***Documents*** є властивістю об'єкта ***Application*** і містить набір об'єктів ***Document***, кожен з яких відповідає відкритому в **Word** документу. Клас Об'єкт ***Document*** містить в якості своїх властивостей сімейства різних об'єктів документа: символів (***Characters***), слів (***Words***), речень (***Sentences***), параграфів (***Paragraphs***), закладок (***Bookmarks***) і т. д.

Об'єктна модель **Excel** побудована за тим же принципом, що і об'єктна модель **Word**. Основним об'єктом, що містить всі інші, є ***Application***. Окремі файли в **Excel** називають робочими книгами. Сімейство ***Workbooks*** в **Excel** є аналогом сімейства ***Documents*** в **Word** і містить набір об'єктів ***Workbook*** (аналог об'єкта ***Document*** в **Word**), кожен з яких відповідає відкритій в **Word** робочій книзі. Нова робоча книга створюється за допомогою методу ***Add*** об'єкта ***Workbooks***.

Для доступу до осередків активного робочого листа **Excel** використовується властивість ***Cells*** об'єкта ***Application***. У загальному випадку це властивість повертає об'єкт ***Range***, який представляє набір комірок. Для отримання або зміни значення окремої комірки застосовується конструкція ***Cells.Item(row,column).Value***, де ***row*** і ***column*** є відповідно номерами рядка і стовпця (починаючи з одиниці), на перетині яких знаходиться ця комірка.

Перепишемо приклади сценаріїв з документами **Word** та **Excel** з ЛР №5 на мові **PowerShell**.

**Файл toWord.ps1**

***$d=Get-Date # отримуємо поточну дату***

***$strText=$d.Day.ToString()+"/" # витягуємо день***

***$strText+=$d.Month.ToString()+"/" # місяць***

***$strText+=$d.Year.ToString()+"`n" # рік***

***$strText+="Доданий рядок зі сценарію PowerShell`n"***

***$oWord=New-Object -ComObject Word.Application # створюємо новий об'єкт Word***

***$oDoc=$oWord.Documents.Add() # створюємо новий документ***

***$oWord.Visible=$true # робимо вікно Word видимим***

***$oDoc.Content.InsertAfter($strText)***

***$oDoc.SaveAs("D:\From PowerShell.doc")***

***$oDoc.Close()***

***$oWord.Quit()***

**Файл toExcel.ps1**

***# створюємо об'єкт - excel-додаток***

***$objXL=New-Object -ComObject Excel.Application***

***# робимо вікно видимим і створюємо робочу книгу***

***$objXL.Visible=$true***

***$objXL.WorkBooks.Add()***

***# встановлюємо ширину першого стовпця***

***$objXL.Columns.Item(1).ColumnWidth=20;***

***# записуємо рядок у комірку (1,1)***

***$objXL.Cells.Item(1,1).Value="Створено зі сценарію PowerShell"***

***Робота з .NET-об'єктами на прикладі Windows.Forms***

Платформа .NET побудована таким чином, що для звернення до тих чи інших об'єктів потрібно попередньо завантажити в операційну пам'ять відповідну збірку (assembly) - динамічну бібліотеку певного виду. Найбільш часто використовуються збірки завантажуються в **PowerShell** автоматично, їх список можна побачити за допомогою статичного методу ***getAssemblies()*** наступним чином:

***PS С:\> [AppDomain]::CurrentDomain.GetAssemblies()***

Наприклад, для звернення до об'єктів ***WinForms*** потрібно за допомогою методу ***LoadWithPartialName*** завантажити збірку, що підтримують ці об'єкти (результат виконання методу приводиться до типу ***void*** для придушення виведення на екран непотрібної інформації):

***PS C:\> [void][System.Reflection.Assembly]::LoadWithPartialName("System.Windows.Forms")***

Наведемо приклад сценарію PowerShell, що відкриває форму, на якій розміщені 2 кнопки (**«ОК»** та **«Cancel»**), поле для введення тексту та надпис для виведення тексту

**CreateForm.ps1**

***#створення нової форми, встановлення її заголовку, ширини та висоти***

***$form=New-Object System.Windows.Forms.Form***

***$form.Text="Перша форма"***

***$form.Width=200***

***$form.Height=200***

***# створення кнопки "ОК"***

***$button1=New-Object System.Windows.Forms.Button***

***$button1.Text="ОК"***

***$button1.Top=20***

***$button1.Left=20***

***$button1.Width=75***

***$button1.Height=23***

***# створення кнопки "Cancel"***

***$button2=New-Object System.Windows.Forms.Button***

***$button2.Text="Cancel"***

***$button2.Top=45***

***$button2.Left=20***

***$button2.Width=75***

***$button2.Height=23***

***# розміщення кнопок на формі***

***$form.Controls.Add($button1)***

***$form.Controls.Add($button2)***

***# створення текстового поля для введення***

***$textbox=New-Object System.Windows.Forms.TextBox***

***$textbox.Top=75***

***$textbox.Left=20***

***$form.Controls.Add($textbox)***

***# створення надпису для відображення тексту***

***$label=New-Object System.Windows.Forms.Label***

***$label.Top=110***

***$label.Left=30***

***$label.Text="Label: "***

***$form.Controls.Add($label)***

Визначимо тепер дію, яка виконуватиметься при натисканні кнопок. Для цього потрібно написати обробник події **Click** кнопки (тобто вказати, які команди повинні виконуватись при натисненні на кнопку). Оброблювач подій - це спеціальний метод з назвою ***Add\_подія***.

***# додавання оброблювачів подій натиснення на кнопки***

***# кнопка "ОК" дублює на надписі введений текст***

***# кнопка "Cancel" закриває форму***

***$button1.Add\_Click(***

***{***

***$text=$textbox.Text***

***$label.Text=$text***

***}***

***)***

***$button2.Add\_Click(***

***{***

***$form.Close()***

***}***

***)***

***# запускаємо форму***

***$form.ShowDialog()***

***Завдання***

1. Написати сценарій **PowerShell**, який аналогічно до скрипта WSH з ЛР№5 приймає введені від користувача рядки, створює нову книгу Excel та документ Word та записує у них введені користувачем рядки (комірки, у які будуть записуватись дані у Excel обрати на власний розсуд). При введенні користувачем слова “Quit” завершити введення даних та зберегти обидва файли.
2. Написати сценарій **PowerShell**, що запускає форму, на якій розміщені 3 кнопки та поле для введення. Кнопки мають надписи «Приховати», «Показати» та «Очистити». Відповідні обробники подій будуть приховувати, показувати поле введення та видаляти у ньому текст.

***Контрольні питання***

1. Як створити новий об’єкт у **PowerShell**? Якими можуть бути дані об’єкти?
2. З чого складається об’єктна модель документів Word та Excel? Як відкрити нові документи Word та Excel у сценаріях **PowerShell**?
3. Які дії потрібно зробити, щоб стало можливим запустити форму зі сценарію **PowerShell**?
4. Як у сценарії **PowerShell** додати візуальні елементи до форми та створити їх обробники подій?