***Лабораторна робота №5***

***«Основи програмування скриптів WSH. Основні методи об’єкта WScript. Запис інформації у документи Word та Excel зі скрипта»***

**Windows Script Host (WSH)** являє собою технологію автоматизації вирішення завдань моніторингу та управління ресурсами в ОС сімейства Microsoft. **WSH** прийшов на зміну мови командних файлів (bat-файлів), вирішуючи аналогічні завдання, але з більш потужною функціональністю.

Сервер сценаріїв **WSH** є розвиненим інструментом, що надає єдиний інтерфейс (об'єктну модель) для спеціалізованих мов, основними з яких є **VBScript**, **Jscript**, але допускається використання додаткових мов, таких як **PerlScript**, **TCL**, **Python** і т. п. Основу функціональних можливостей технології **WSH** складають компоненти **ActiveX**, сервісні функції яких дозволяють управляти ресурсами системи (створювати ярлики програм, вимикати комп'ютер, змінювати записи в реєстрі, працювати з мережею і користувачами, виводити список дисків, підключати/відключати мережеві диски, отримувати ім'я комп'ютера і користувача, працювати зі змінними оточення, видавати діалогові та інформаційні повідомлення тощо), а також управляти роботою інших додатків (серверів автоматизації), наприклад Microsoft Word і чи Micosoft Excel.

При цьому для роботи з **WSH** використовуються мови програмування **VBScript** і **JScript**, підтримка яких вбудована в операційну систему і не вимагає установки спеціалізованих систем програмування. Середовищем програмування фактично стає текстовий редактор, наприклад, стандартний **Блокнот**. Типовий WSH-сценарій - це звичайний текстовий файл з розширенням ***.js*** (для **JScript**) або ***.vbs*** (для **VBScript**) відповідно.

Мова **JScript** - скриптова, об'єктно-орієнтована мова програмування, що інтерпретується, розроблена фірмою Microsoft. Спочатку вона використовувалась для створення динамічних HTML-сторінок в рамках технології ASP. Синтаксично вона дуже близька до мов **Java** і **C**, по-цьому є хорошим вибором для написання скриптів тими програмістами, які володіють цими мовами. Мова **VBScript (Visual Basic Script Edition)** - являє собою полегшену версію мови **Microsoft Visual Basic**, і вона буде хорошим вибором для тих, хто володіє **Visual Basic** або **VBA (Visual Basic for Application)**.

Для запуску WSH-скрипта використовуються команди ***cscript.exe*** і ***wscript.exe***. Перша з них направляє виведення скрипта на стандартну консоль виведення в текстовий режим, а друга - в діалогове вікно. У цьому можна переконатися, якщо запустити наступний скрипт на мові **VBScript**.

***'Найпростіший скрипт на VBScript***

***WScript.Echo "Привіт!"***

командами

***cscript.exe hello.vbs*** та ***wscript.exe hello.vbs***.

Аналогічний по функціональності скрипт на **JScript** виглядає наступним чином:

***//Найпростіший скрипт на JScript***

***WScript.Echo("Привіт!");***

При запуску сценарію з командного рядка можна визначити параметри хоста (див. Табл. 1), які включають або вимикають різні опції **Windows Scripting Host** і завжди передуються двома слешами (***//***).

Табл. 1. Параметри командного рядка для ***cscript.exe***

|  |  |
| --- | --- |
| **Параметр** | **Опис** |
| ***//B*** | Включає пакетний режим. При цьому на екран будуть виводитися всі повідомлення про помилки в сценарії |
| ***//T:nn*** | Задає таймаут в секундах. Сценарій виконується не більше nn секунд |
| ***//I*** | Вимикає пакетний режим (використовується за умовчанням). На екран не виводяться повідомлення про помилки |
| ***//logo*** | Включає в висновок скрипта напис про версії та розробника WSH (використовується за умовчанням) |
| ***//nologo*** | Пригнічує інформацію про версії та розробника **WSH** |
| ***//H: Cscript or Wscript*** | Робить **CSCRIPT.EXE** або **WSCRIPT.EXE** хостом за замовчуванням, тобто асоціює їх з розширеннями скриптів |
| ***//S*** | Зберігає установки командного рядка для поточного користувача |
| ***//D*** | Функція переходу в режим Active Debuging |
| ***//X*** | Виконання сценарію у відлагоджувачі |

Безсумнівною перевагою WSH в порівнянні з командними файлами є підтримка об'єктно-орієнтованої парадигми, коли основні об'єкти предметної області (такі як файл на диску, ярлик на робочому столі, змінна оточення, запущений процес і ін.) представлені програмними об'єктами. Маніпулювання властивостями і методами цих об'єктів дозволяє швидко і ефективно отримати або змінити властивості об'єкта.

За допомогою внутрішніх об'єктів WSH з сценаріїв можна виконувати основні завдання:

* виводити інформацію в стандартний вихідний потік (на екран) або у вікно Windows;
* читати дані з стандартного вхідного потоку (тобто. вводити дані з клавіатури) або використовувати інформацію, виведену іншою командою;
* використовувати властивості і методи зовнішніх об'єктів, а також обробляти події цих об'єктів;
* запускати нові процеси або активізувати вже наявні;
* працювати з локальною мережею: визначати ім'я зареєстрованого користувача, підключати мережеві диски та принтери;
* переглядати і змінювати змінні середовища;
* отримувати доступ до спеціальних папках Windows;
* створювати ярлики Windows;
* працювати з системним реєстром.

До WSH входять наступні об'єкти:

1. ***WScript***. Це головний об'єкт WSH, який служить для створення інших об'єктів або зв'язку з ними, містить відомості про сервер сценаріїв, а також дозволяє вводити дані з клавіатури і виводити інформацію на екран або у вікно Windows;

2. ***WshArguments***. Забезпечує доступ до параметрів командного рядка за-пущеного сценарію або ярлика Windows;

3. ***WshEnvironment***. Призначений для роботи зі змінними середовища;

4. ***WshSpeciaiFoiders***. Забезпечує доступ до спеціальних папок Windows;

5. ***WshNetwork***. Використовується при роботі з локальною мережею: містить мережеву інформацію для локального комп'ютера, дозволяє підключати мережеві диски та принтери;

6. ***WshShell***. Дозволяє запускати процеси, створювати ярлики, працювати зі змінними середовища, системним реєстром і спеціальними папками Windows;

7. ***WshShortcut***. Дозволяє працювати з ярликами Windows;

8. ***WshUriShortcut***. Призначений для роботи з ярликами мережевих ресурсів.

Крім цього, є об'єкт ***Scripting.FileSystemObject***, що забезпечує доступ до файлової системи комп'ютера.

Одне з центральних місць в скриптах **WSH** займає об'єкт ***WScript***. Об'єкт ***WScript*** можна використовувати в сценарії **WSH** відразу, без будь-якого попереднього опису або створення, тому що його екземпляр створюється сервером сценаріїв (***CScript.exe*** або ***WScript.exe***) автоматично. Об'єкт ***WScript*** дозволяє здійснювати операції введення-виведення інформації, отримувати список аргументів скрипта, отримувати інформацію про сервер **WSH**, а також дозволяє скриптам запускати додатки і управляти ними.

Табл. 2. Властивості та методи об’єкта ***WScript***

|  |  |
| --- | --- |
| **Властивість** | **Опис** |
| ***Arguments*** | Повертає покажчик на колекцію ***WshArguments*** аргументів командного рядка |
| ***FullName*** | Повертає ім'я виконуваного файлу хоста і повний шлях до нього (наприклад, ***C:\Windows\wscript.exe***) |
| ***Name*** | Виводить назву сервера об'єкта: **Windows Scripting Host** |
| ***Path*** | Визначає каталог і шлях, що містять ***wscript.exe*** або ***cscript.exe*** |
| ***ScriptFullName*** | Повертає повний шлях і ім'я виконуваного в даний момент скрипта |
| ***ScriptName*** | Те ж, що і ***ScriptFullName***, але без шляху |
| ***StdErr*** | Дозволяє запущеному сценарієм записувати повідомлення в стандартний потік для помилок |
| ***StdIn*** | Дозволяє запущеному сценарію читати інформацію з стандартного вхідного потоку |
| ***StdOut*** | Дозволяє запущеному сценарію записувати інформацію в стандартний вихідний потік |
| ***Version*** | Повертає версію встановленого **Windows Scripting Host** |
| **Метод** | **Опис** |
| ***CreateObject*** | Створює об'єкт по його **ProgID** |
| ***ConnectObject*** | Дозволяє підключитися до подій об'єкта. Як параметр приймає об'єкт, до якого потрібно підключитись, і префікс відповідних подій процедур, реалізованих в скрипті |
| ***DisconnectObject*** | Вимикає від об'єкта, підключеного попереднім методом |
| ***Echo*** | Виводить текстовий рядок (в ***cscript*** - в **StdOut**, в ***wscript*** - у вигляді діалогового вікна) |
| ***GetObject*** | Дозволяє отримати покажчик на об'єкт з файлу або об'єкта, наданого в секції **strProgID** |
| ***Quit*** | Завершує скрипт |
| ***Sleep*** | Переводить скрипт в неактивний стан на час, вказаний в мілісекундах |

Необхідно відзначити, що стандартні потоки введення-виведення доступні в об'єкті ***WScript*** через властивості ***StdIn***, ***StdOut*** і ***StdErr***, тільки якщо сценарій запускався в консольному режимі за допомогою ***cscript.exe***.

Розглянемо простий скрипт на мові JScript, що використовує властивості і методи об'єкта ***WScript***:

***var args;***

***args=WScript.Arguments;***

***for (var i=0; i<args.length; i++) //можна i<args.Count()***

***WScript.Echo("Аргумент "+i+": "+args(i));***

Запустити цей скрипт (припустимо, що він записаний в файл ***arglist.js***) можна з командного рядка:

***cscript.exe arglist.js /a myFolder \*.txt***

Можна запустити і скрипт з віконним інтерфейсом:

***wscript.exe arglist.js /a myFolder \*.txt***

але кожен виклик методу ***Echo*** буде створювати окреме вікно повідомлення.

Якщо встановлена за замовчуванням асоціація розширення ***js*** (і ***vbs***) з програмою ***cscript.exe*** не була змінена, то скрипт запускається простіше:

***arglist.js /a myFolder \*.txt***

Якщо створений консольний скрипт, то для введення-виведення можна використовувати методи ***Write, WriteLine, Read, ReadLine, ReadAll, Skip, SkipLine*** властивостей-об'єктів ***StdIn, StdOut і StdErr***. Наступний приклад запитує у користувача повне ім'я файлу і перевіряє, чи існує такий файл на диску:

***var FileName, fso;***

***WScript.StdOut.WriteLine("Введіть повне ім'я файлу ");***

***FileName=WScript.StdIn.ReadLine();***

***fso = WScript.CreateObject("Scripting.FileSystemObject");***

***if (fso.FileExists(FileName))***

***WScript.StdOut.WriteLine("Такий файл вже існує ");***

***else***

***WScript.StdOut.WriteLine("Такий файл не існує ");***

Виведення інформації з скрипта в вихідний потік дозволяє перенаправляти інформацію в файл або організовувати конвеєр стандартними операторами операційної системи (***>***, ***>>***, ***|***):

***cscript.exe someScript.js | find “2016” > lastYear.txt***

З уже розглянутих прикладів очевидно, що головний секрет успішного застосування **WSH** полягає в знанні та вмінні маніпулювати властивостями і методами перерахованих в табл. 1 об'єктів. Деякі з цих об'єктів створюються автоматично при ініціалізації властивості інших об'єктів (наприклад, вже розглянутий нами раніше ***WshArguments***), інші необхідно створювати в коді. Для цього служить метод ***CreateObject*** об'єкта ***WScript***. Наприклад, для створення об'єкта ***WshShell*** необхідно виконати:

***var wh=WScript.CreateObject("WScript.Shell");***

Після виконання цього методу буде створений об'єкт ***WshShell***, доступ до властивостей і методів якого можна за допомогою змінної ***wh***.

Як параметр метод ***CreateObject*** приймає програмний ідентифікатор об'єкта (**Programmic Identifier, ProgID**). **ProgID** є унікальним ідентифікатором об'єкта, його зв'язок з конкретним розташуванням файлів, що зберігають код об'єкта, здійснюється за допомогою технології **COM** через системний реєстр. Зокрема, так можна зв'язати скрипт з екземпляром **Microsoft Excel** і записати дані в осередку аркуша документа:

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

***var objXL = WScript.CreateObject("Excel.Application");***

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

***objXL.Visible = true;***

***objXL.WorkBooks.Add();***

***//*** ***встановлюємо ширину першого стовпця***

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

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

***objXL.Cells(1, 1).Value = "Створено з WSH";***

Наступний приклад демонструє, як можна дописати інформацію в кінець файлу типу ***doc*** (документ **Microsoft Word**):

***var strDoc = "c:\\test.doc";***

***var d = new Date(); //*** ***отримуємо поточну дату***

***var strText;***

***strText += d.getDate() + "/"; //*** ***витягуємо день,***

***strText += (d.getMonth() + 1) + "/"; //місяць***

***strText += d.getYear(); //рік***

***strText += " Доданий рядок з WSH скрипта\n" ;***

***if (WScript.CreateObject("Scripting.FileSystemObject").***

***FileExists(strDoc))***

***{ var oWord=WScript.CreateObject("Word.Application");***

***var oDoc=oWord.Documents.Open(strDoc);***

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

***oDoc.Save();***

***oDoc.Close();***

***oWord.Quit();***

***}***

***else***

***{***

***WScript.Echo("Document [" + strDoc + "] not found");***

***WScript.Quit(1);***

***}***

***WScript.Quit(0);***

Отримати ProgID і список методів і властивостей для цікавить вас об'єкта можна з технічної документації або використовуючи бібліотеку його типів.

***Завдання***

1. Написати скрипт, що виводить текстове повідомлення методами WScript.Echo і WshShell.Popup. Запустіть скрипт командами cscript і wscript, порівняйте результати виведення.
2. Написати скрипт, що 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.

***Контрольні питання***

1. Для чого призначений об’єкт WScript?
2. За допомогою яких властивостей об’єкта WScript можна отримати доступ до стандартних потоків введення, виведення та помилок?
3. Для чого призначений метод CreateObject об’єкта WScript?
4. Для чого призначений метод ConnectObject об’єкта WScript?
5. Як зі сценарію WSH відкрити файл електронної таблиці Microsoft Excel та записати інформацію у її комірках?
6. Як зі сценарію WSH відкрити файл документу Microsoft Word та записати інформацію у ньому?