Сервер сценариев WSH. Языки сценариев BScript и JScript

Шестаков А.П.

Введение

Был рассмотрен язык командных файлов (язык командной оболочки), который в качестве инструмента для автоматизации работы поддерживается во всех версиях *Windows*. Однако с помощью командного интерпретатора cmd.exe трудно написать какую-либо сложную программу-*сценарий* (*script*): отсутствует полноценная *интерактивность*, нельзя напрямую работать с рабочим столом *Windows* и системным реестром и т. д.

Для исправления этой ситуации компанией Microsoft был разработан *сервер* сценариев *Windows Script Host* (WSH), с помощью которого можно выполнять сценарии, написанные, в принципе, на любом языке (при условии, что для этого языка установлен соответствующий *модуль* (*scripting engine*), поддерживающий технологию ActiveX Scripting). В качестве стандартных языков поддерживаются *Visual Basic Script Edition* (VBScript) и JScript.

Вообще говоря, принцип работы сценариев, поддерживаемых WSH, состоит в использовании объектов ActiveX

Введение

Сервер сценариев WSH является мощным инструментом, предоставляющим единый интерфейс (объектную модель) для специализированных языков (VBScript, JScript, PerlScript, *REXX*, TCL, Python и т. п.), которые, в свою очередь, позволяют использовать любые внешние объекты ActiveX. С помощью WSH сценарии могут быть выполнены непосредственно в операционной системе Windows, без встраивания в HTML-страницы.

Назначение WSH

Некоторые задачи, для автоматизации которых прекрасно подходят сценарии WSH.

- Организация резервного копирования на сетевой сервер файлов с локальной машины, которые отбираются по какому-либо критерию.
- Быстрое изменение конфигурации рабочего стола Windows в зависимости от задач, выполняемых пользователем.
- Автоматический запуск программ Microsoft Office, создание там сложных составных документов, распечатка этих документов и закрытие приложений.
- Управление работой приложений, не являющихся серверами автоматизации, с помощью посылки в эти приложения нажатий клавиш.
- Подключение и отключение сетевых ресурсов (дисков и принтеров).
- Создание сложных сценариев регистрации для пользователей.
- Выполнение задач администрирования локальной сети (например, добавление или удаление пользователей).

Запуск скрипта

- Консольный режим

  cscript имя_файла
- Оконный режим

  wscript имя_файла

VBScript vs Jscript!

Внутренние объекты WSH

С помощью внутренних объектов WSH из сценариев можно выполнять следующие основные задачи:

- выводить информацию в стандартный выходной поток (на экран) или в диалоговое окно Windows;
- читать данные из стандартного входного потока (то есть вводить данные с клавиатуры) или использовать информацию, выводимую другой командой;
- использовать свойства и методы внешних объектов, а также обрабатывать события, которые генерируются этими объектами;
- запускать новые независимые процессы или активизировать уже имеющиеся;
- запускать дочерние процессы с возможностью контроля их состояния и доступа к их стандартным входным и выходным потокам;
- работать с локальной сетью: определять имя зарегистрировавшегося пользователя, подключать сетевые диски и принтеры;
- просматривать и изменять переменные среды;
- получать доступ к специальным папкам Windows;
- создавать ярлыки Windows;
- работать с системным реестром.

Внутренние объекты WSH

- **WScript**. Это главный объект WSH, который служит для создания других объектов или связи с ними, содержит сведения о сервере сценариев, а также позволяет вводить данные с клавиатуры и выводить информацию на экран или в окно Windows.
- **WshArguments**. Обеспечивает доступ ко всем параметрам командной строки запущенного сценария или ярлыка Windows.
- **WshNamed**. Обеспечивает доступ к именным параметрам командной строки запущенного сценария.
- **WshUnnamed**. Обеспечивает доступ к безымянным параметрам командной строки запущенного сценария.
- **WshShell**. Позволяет запускать независимые процессы, создавать ярлыки, работать с переменными среды, системным реестром и специальными папками Windows.
- **WshSpecialFolders**. Обеспечивает доступ к специальным папкам Windows.
- **WshShortcut**. Позволяет работать с ярлыками Windows.
- **WshUrlShortcut**. Предназначен для работы с ярлыками сетевых ресурсов.

Внутренние объекты WSH

- **WshEnvironment**. Предназначен для просмотра, изменения и удаления переменных среды.
- **WshNetwork**. Используется при работе с локальной сетью: содержит сетевую информацию для локального компьютера, позволяет подключать сетевые диски и принтеры.
- **WshScriptExec**. Позволяет запускать консольные приложения в качестве дочерних процессов, обеспечивает контроль состояния этих приложений и доступ к их стандартным входным и выходным потокам.
- **WshController**. Позволяет запускать сценарии на удаленных машинах.
- **WshRemote**. Позволяет управлять сценарием, запущенным на удаленной машине.
- **WshRemoteError**. Используется для получения информации об ошибке, возникшей в результате выполнения сценария, запущенного на удаленной машине.

Объект WScript

Свойства объекта WScript позволяют получить полный путь к использующемуся серверу сценариев (wscript.exe или cscript.exe), параметры командной строки, с которыми запущен сценарий, режим его работы (интерактивный или пакетный). Кроме этого, с помощью свойств объекта WScript можно выводить информацию в стандартный выходной поток и читать данные из стандартного входного потока. Также WScript предоставляет методы для работы внутри сценария с объектами автоматизации и вывода информации на экран (в текстовом режиме) или в окно Windows.

Отметим, что в сценарии WSH объект WScript можно использовать сразу, без какого-либо предварительного описания или создания, так как его экземпляр создается сервером сценариев автоматически. Для использования же всех остальных объектов нужно использовать либо метод CreateObject, либо определенное свойство другого объекта.

Методы объекта WScript

Метод	Описание
`CreateObject(strProgID [, strPrefix])`	Создает объект, заданный параметром `strProgID`
`ConnectObject(strObject, strPrefix)`	Устанавливает соединение с объектом `strObject`, позволяющее писать функции-обработчики его событий (имена этих функций должны начинаться с префикса `strPrefix`)
`DisconnectObject(obj)`	Отсоединяет объект `obj`, связь с которым была предварительно установлена в сценарии
`Echo([Arg1] [, Arg2] [,…])`	Выводит текстовую информацию на консоль или в диалоговое окно
`GetObject(strPathname [, strProgID], [strPrefix])`	Активизирует объект автоматизации, определяемый заданным файлом (параметр `strPathName`) или объект, заданный параметром `strProgID`
`Quit([intErrorCode])`	Прерывает выполнение сценария с заданным параметром `intErrorCode` кодом выхода. Если параметр `intErrorCode` не задан, то объект WScript установит код выхода равным нулю
`Sleep(intTime)`	Приостанавливает выполнения сценария (переводит его в неактивное состояние) на заданное параметром `intTime` число миллисекунд

Метод CreateObject

Строковый параметр strProgID, указываемый в методе CreateObject, называется программным идентификатором объекта (Programmic Identifier, ProgID).

Если указан необязательный параметр strPrefix, то после создания объекта в сценарии можно обрабатывать события, возникающие в этом объекте (естественно, если объект предоставляет интерфейсы для связи с этими событиями). Когда объект сообщает о возникновении определенного события, сервер сценариев вызывает функцию, имя которой состоит из префикса strPrefix и имени этого события. Например, если в качестве strPrefix указано "MYOBJ_", а объект сообщает о возникновении события "OnBegin," то будет запущена функция "MYOBJ_OnBegin", которая должна быть описана в сценарии.

В примере метод CreateObject используется для создания объекта WshNetwork:

```
var WshNetwork = WScript.CreateObject("WScript.Network");
```

Объект WshShell

С помощью объекта WshShell можно

- запускать новый процесс
- создавать ярлыки
- работать с системным реестром
- получать доступ к переменным среды и специальным папкам Windows.

Создается этот объект следующим образом:

```
var WshShell=WScript.CreateObject("WScript.Shell");
```

Метод	Описание
AppActivate(title)	Активизирует заданное параметром title окно приложения. Строка title задает название окна (например, "calc" или "notepad") или идентификатор процесса (Process ID, PID)
CreateShortcut(strPathname)	Создает объект WshShortcut для связи с ярлыком Windows (расширение lnk) или объект WshUrlShortcut для связи с сетевым ярлыком (расширение url). Параметр strPathname задает полный путь к создаваемому или изменяемому ярлыку
Environment(strType)	Возвращает объект WshEnvironment, содержащий переменные среды заданного вида
Popup(strText,[nSecToWait], [strTitle], [nType])	Выводит на экран информационное окно с сообщением, заданным параметром strText. Параметр nSecToWait задает количество секунд, по истечении которых окно будет автоматически закрыто, параметр strTitle определяет заголовок окна, параметр nType указывает тип

RegDelete(strName)	Удаляет из системного реестра заданный параметр или раздел целиком
RegRead(strName)	Возвращает значение параметра реестра или значение по умолчанию для раздела реестра
RegWrite(strName, anyValue [,strType])	Записывает в реестр значение заданного параметра или значение по умолчанию для раздела
Run(strCommand, [intWindowStyle], [bWaitOnReturn])	Создает новый независимый процесс, который запускает приложение, заданное параметром strCommand
SendKeys(string)	Посылает одно или несколько нажатий клавиш в активное окно (эффект тот же, как если бы вы нажимали эти клавиши на клавиатуре)
SpecialFolders(strSpecFolder)	Возвращает строку, содержащую путь к специальной папке Windows, заданной параметром strSpecFolder

Объекты для основных операций с файловой системой

Для работы с файловой системой из сценариев WSH предназначены восемь объектов, главным из которых является **FileSystemObject**. С помощью методов объекта **FileSystemObject** можно выполнять следующие основные действия:

- копировать или перемещать файлы и каталоги;
- удалять файлы и каталоги;
- создавать каталоги;
- создавать или открывать текстовые файлы;
- создавать объекты Drive, Folder и File для доступа к конкретному диску, каталогу или файлу соответственно.

С помощью свойств объектов Drive, Folder и File можно получить детальную информацию о тех элементах файловой системы, с которыми они ассоциированы. Объекты Folder и File также предоставляют методы для манипулирования файлами и каталогами (создание, удаление, копирование, перемещение); эти методы в основном копируют соответствующие методы объекта **FileSystemObject**

Получение сведений об определенном диске (тип файловой системы, метка тома, общий объем и количество свободного места и т.д.)	Свойства объекта `Drive`. Сам объект `Drive` создается с помощью метода `GetDrive` объекта `FileSystemObject`
Получение сведений о заданном каталоге или файле (дата создания или последнего доступа, размер, атрибуты и т.д.)	Свойства объектов `Folder` и `File`. Сами эти объекты создаются с помощью методов `GetFolder` и `GetFile` объекта `FileSystemObject`
Проверка существования определенного диска, каталога или файла	Методы `DriveExists`, `FolderExists` и `FileExists` объекта`FileSystemObject`
Копирование файлов и каталогов	Методы `CopyFile` и `CopyFolder` объекта `FileSystemObject`, а также методы `File.Copy` и `Folder.Copy`
Перемещение файлов и каталогов	Методы `MoveFile` и `MoveFolder` объекта `FileSystemObject`, или методы `File.Move` и `Folder.Move`
Удаление файлов и каталогов	Методы `DeleteFile` и `DeleteFolder` объекта`FileSystemObject`, или методы `File.Delete` и `Folder.Delete`

Создание каталога	Методы `FileSystemObject.CreateFolder` или `Folders.Add`
Создание текстового файла	Методы `FileSystemObject.CreateTextFile` или `Folder.CreateTextFile`
Получение списка всех доступных дисков	Коллекция `Drives`, содержащаяся в свойстве `FileSystemObject.Drives`
Получение списка всех подкаталогов заданного каталога	Коллекция `Folders`, содержащаяся в свойстве `Folder.SubFolders`
Получение списка всех файлов заданного каталога	Коллекция `Files`, содержащаяся в свойстве `Folder.Files`
Открытие текстового файла для чтения, записи или добавления	Методы `FileSystemObject.CreateTextFile` или `File.OpenAsTextStream`
Чтение информации из заданного текстового файла или запись ее в него	Методы объекта `TextStream`