# Лабораторная работа

# Изучение возможностей сервера сценариев WSH

Сервер сценариев WSH (Windows Script Host) 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 среды WSH сценарии могут быть выполнены непосредственно в операционной системе Windows, без встраивания в HTML-страницы.

Целью лабораторной работы является изучение возможностей сервера WSH при разработке сценариев, способов их использования и приобретение навыков в написании и отладке.

Конкретные задания на работу выделены в тексте *курсивом*.

## 1. Вводные замечания

WSH позволяет работать с файловой системой, создавать ярлыки программ, выключать компьютер, изменять (добавлять и удалять) записи в реестре, работать с сетью и пользователями (выводить список дисков, подключать/отключать сетевые диски, получать имя компьютера и пользователя), работать с переменными окружения, выдавать диалоговые и информационные сообщения и многое другое. С помощью wsh-скриптов можно управлять запуском программ, посылать сигналы другим процессам, удаленно администрировать систему, работать с сетевым принтером, управлять входом в систему (login-скрипты) и многое другое. Скрипты для Windows Script Host могут быть не только отдельными программами, но и встраиваться в HTML-страницы.

Для написания WSH-скрипта можно использовать любой текстовый редактор. Файл с готовым скриптом на VBScript должен иметь расширение vbs. Чтобы запустить готовый скрипт, достаточно дважды щелкнуть мышкой по файлу, либо ввести полное имя файла скрипта в «Пуск» и «Выполнить».

## 2. Объектная модель WSH

Для того чтобы воспользоваться всеми возможностями, которые представляет эта технология, нужно хорошо разобраться в ее структуре. Windows Script Host состоит из 14 объектов. Самым главным объектом является объект **WScript**.

Рассмотрим основные объекты и их возможности:

**1) WScript**

- Устанавливает или получает аргументы командной строки.
- Определяет имя скриптового файла.
- Определяет имя хоста для скрипта (wscript.exe или cscript.exe).
- Определяет версию хоста.
- Создает, соединяется или отсоединяется от объектов COM.
- Программно останавливает выполнение скрипта.
- Выводит сообщения.

**2) WshArguments**
Получает доступ к аргументам командной строки.
**3) WshNamed**
Получает доступ к именованным аргументам командной строки.
**4) WshUnnamed**
Получает доступ к безымянным аргументам командной строки.
**5) WshNetwork**
Работа с сетью.
**6) WshController**
Для работы со скриптами удаленного управления.
**7) WshRemote**
Удаленные скрипты.
**8) WshRemote Error**
Информация об ошибках (для WshRemote).
**9) WshShell**
Работа с оболочкой Windows.
**10) WshShortcut**
Создание ярлыков.
**11) WshSpecialfolders**
Пути к специальным папкам Windows.
**12) WshURLShortcut**
Создание интернет-ссылок.
**13) WshEnvironment**
Доступ к коллекции переменных окружения.
**14) WshScriptExec**
Работа со скриптами.

Наряду с перечисленными объектами, в полной мере может быть использован объект FileSystemObject, который был частично изучен в лабораторной работе «Использование скриптов в среде WSH».

Все объекты имеют свои свойства и методы, в табл.1 приведены свойства для главного объекта **Wscript**.

Таблица 1. Описание свойств объекта Wscript.

| Свойство | Описание |
|---|---|
| Arguments | Возвращает указатель на список аргументов командной строки |
| FullName | Возвращает имя исполняемого файла хоста и полный путь к нему |
| Name | Выводит надпись Windows Script Host |
| Path | Определяет каталог и путь, содержащие wscript.exe или cscript.exe |
| ScriptFullName | Возвращает полный путь и имя исполняемого в данный момент скрипта |
| ScriptName | То же, что и ScriptFullName, но без пути |
| Version | Возвращает версию установленного Windows Script Host |

В качестве первого примера рассмотрим, как WSH поддерживает Drag'n'Drop события. С помощью свойства Arguments объекта WScript в своем скрипте можно просто получать имена файлов, с которыми надо работать. Для этих целей можно использовать скрипт из примера 1.1.

Пример 1.1. Поддержка Drag'n'Drop.

```
'Полное имя файла
Set objArgs = WScript.Arguments
For i = 0 to objArgs.Count - 1
   WScript.Echo objArgs(i)           ' вывод сообщения
Next
```

***Задание 1.***

*Вставьте скрипт примера 1.1 в блокнот и сохраните под именем drag_n_drop.vbs. Теперь просто перетащите любой файл (папку), разместив его над скриптом drag_n_drop.vbs,* и отпустите клавишу мыши. Вы тут же увидите полное имя этого файла (папки). Объясните назначение свойства «.Count».

Как видно из этого простого примера объекты WSH обладают действительно очень интересными возможностями. В примере 1.2 представлен скрипт, который выводит на экран сообщения, касающиеся свойств WScript приведенных в табл.1.

Пример 1.2. Работа со свойствами WScript.

```
'Изучаем свойства WScript
WScript.Echo WScript.FullName
WScript.Echo WScript.Name
WScript.Echo WScript.Path
WScript.Echo WScript.ScriptFullName
WScript.Echo WScript.ScriptNamесвойств и методов
WScript.Echo WScript.Version
```

***Задание 2.***

*Запустите скрипт примера 1.2.* Объясните смысл всех появляющихся сообщений.

## 2. Обеспечение доступа к оборудованию компьютера

Ниже приведен пример скрита, осуществляющего непосредственное взаимодействие с оборудованием (hardware) компьютера. В примере 2.1 к нулевому пункту `item(0)` свойства `cdromCollection` применен метод `Eject`.

Пример 2.1. Открытие CD дисковода

```
'Открываем дисковод
CreateObject("WMPlayer.OCX.7").cdromCollection.item(0).Eject
```

Анализ адресного пространства внешней памяти можно осуществить с использованием объекта FileSystemObject. Ниже, в примере 2.2 приведен скрипт анализирующий наличие свободного адресного пространства на трех дисках.

Пример 2.2. Анализ внешнего адресного пространства

```
'Проверка количества свободного места на дисках
Set fso = WScript.CreateObject("Scripting.FileSystemObject")
Set WSHShell = WScript.CreateObject("WScript.Shell")
'Проверяем все драйвы (HDD, FDD, CDD) в системе
For each i In fso.Drives
  If i.DriveType=2 Or i.DriveType=1 Or i.DriveType=0 Then
    'Получаем букву диска
    drive=i.DriveLetter
```

```
    'Узнаем свободное место и переводим его в Мб с
    free = FormatNumber(fso.GetDrive(drive).FreeSpace/1048576,0)
    WSHShell.Popup("На диске "+drive+" свободно "+free+" Мбайт")
  End If
Next
```

Если не фильтровать тип диска (строка `If i.DriveType=2 …`), то объем свободного места будет проверяться и на дискете в дисководе и на CD-ROM'е. А если в приводе не будет носителя, то скрипт прервется и будет выдано сообщение об ошибке.

DriveType имеет следующие значения:
**0** - Тип не может быть определен (флэш память на USB)
**1** - Сменный носитель или дисковод для гибких дисков
**2** - Обычный HDD
**3** - Сетевой диск
**4** - CD-ROM
**5** - Виртуальный RAM-диск

Остановимся подробнее на строке:

```
free = FormatNumber(fso.GetDrive(drive).FreeSpace /1048576, 0)
```.

Свойство FreeSpace возвращает количество свободного дискового места в байтах. Конечно, работать с такими большими цифрами некомфортно, поэтому мы переводим байты в мегабайты, делением на 1048576. А чтобы не лицезреть кучу знаков после запятой, округляем их до нуля с помощью `FormatNumber()` - количество знаков после запятой устанавливается с помощью второй переменной.

Если надо получить полный объем диска, то вместо FreeSpace используется свойство `TotalSize`.

Для того, чтобы явно указать диск, с которым необходимо работать используется метод `GetDrive`:

```
Set fso = WScript.CreateObject("Scripting.FileSystemObject")
Set Drive = fso.GetDrive("c")
```

Остальные свойства и методы работы с диском не так интересны. Среди них такие как:

`DriveExists` - проверяет на наличие заданного диска: *`fso.DriveExists("c")`*.
`IsReady` - если диск готов к использованию, возвращает true, в противном случае - false.
`RootFolder` - возвращает путь к корневому каталогу.
`SerialNumber` - возвращает серийный номер диска.
`ShareName` - возвращает сетевое имя диска.
`VolumeName` - возвращает либо устанавливает метку диска.

## ***Задание 3.***

*Напишите скрипт определяющий полный объем, свободное адресное пространство и серийные номера всех HDD дисков, установленных на компьютере. Вся информация должна быть получена в одном сообщении.*

## 3. Работа с файловой системой

Чтобы узнать тип файловой системы любого диска используется свойство `FileSystem`. Пример 3.1 показывает как используется это свойство в скриптах.

Пример 3.1. Определение типа файловой системы диска C

```
'Тип файловой системы
Set fso = WScript.CreateObject("Scripting.FileSystemObject")
Set Drive = fso.GetDrive("c")
MsgBox(Drive.FileSystem)
```

Теперь обратим внимание на работу с файлами и папками. Редактирование файлов было рассмотрено в лабораторной работе «Использование скриптов в среде WSH», а здесь коснемся вопросов создания, копирования, удаления файлов, установки атрибутов и т.п. Для примера создадим на диске D:\ папку TEST, а в ней файл test.txt.

Пример 3.2. Создание директория и файла

```
//Создание папки и файла на языке JavaScript
var fso = new ActiveXObject("Scripting.FileSystemObject");
//Проверяем, есть ли такая папка
if (fso.FolderExists("D:\\TEST"))
   WScript.Echo("Такая папка уже существует");
else
//Если нет - создаем
   var Folder = fso.CreateFolder("D:\\TEST");
//Проверяем, есть ли уже такой файл
if (fso.FileExists("D:\\TEST\\test.txt"))
   WScript.Echo("Такой файл уже существует");
else
//Если нет, то создаем его
   var File = fso.CreateTextFile("D:\\TEST\\test.txt", true);
```

Комментарии наглядно демонстрируют работу скрипта. Подробнее остановимся на создании файла (последняя строка). Вторым параметром передается значение `true`. Этот параметр является необязательным и указывает, перезаписывать ли файл с таким именем. Здесь же может передаваться и третий параметр. Если он отсутствует или равен `false`, то файл будет создан в кодировке ASCII. Если параметр равен `true` - в unicode.

В примере 3.2 создана не только сама папка и файл, но и объекты этой папки и файла (`Folder` и `File` соответственно). Если надо создать объект для уже существующей папки / файла, используется метод `GetFile/GetFolder`:

```
var File1 = fso.GetFile("C:\\autoexec.bat");
```

К этим объектам применимы различные методы, позволяющие копировать, переименовывать, удалять файлы, получать о них различные сведения, устанавливать атрибуты. Как несложно догадаться для копирования, перемещения и удаления файлов и папок применяются методы `Copy`, `Move` и `Delete`. При этом в `Copy` и `Move` передается имя файла, в который надо скопировать/перенести исходный, а в `Delete` ничего не передается. Так, чтобы скопировать autoexec.bat в autoexec.tmp надо добавить строку

```
File1.Copy("C:\\autoexec.tmp");
```

Различные свойства объектов, созданных с помощью `GetFile/GetFolder` позволяют получить различные сведения о файлах и папках на которые они указывают. Вот эти свойства:

`Size` - возвращает размер файла/папки;
`DateCreated` - время создания;
`DateLastAccessed` - время последнего обращения к объекту;
`DateLastModified` - время последнего изменения.

Например:

```
WScript.Echo(File1.Size);
```

Бывают ситуации, когда надо получить имя файла, на который ссылается объект. Например:

```
var fso = new ActiveXObject("Scripting.FileSystemObject");
var File1 = fso.GetFile("C:\\autoexec.bat");
var File2 = fso.GetFile("C:\\config.sys");
var File3 = fso.GetFile("C:\\netlog.txt");
if (fso.FileExists("C:\\config.sys"))
   File1 = File2;
else
   File1 = File3;
WScript.Echo (File1.Name);
```

В зависимости от того, есть ли файл `config.sys`, объект `File1` будет указывать на разные файлы. Конечно, в таком виде этот пример в реальной ситуации вряд ли встретится, но для иллюстрации неопределенности с файлом вполне подходит. Для определения имени файла (папки) и пути к нему служат еще несколько свойств:

`Name` - возвращает обычное имя файла (папки);
`ShortName` - короткое имя (в формате MS-DOS). Длинные имена будут урезаны до формата 8.3;
`Path` - возвращает обычный путь к файлу (папке);
`ShortPath` - короткий путь (с тильдой "~");
`ParentFolder` - возвращает имя родительского каталога.

Пример 3.3. Анализ папок (директориев) на диске D.

```
'Анализ диска
Dim fso, f, f1, fc, s
   Set fso = CreateObject("Scripting.FileSystemObject")
   disk = "d:"
   Set f = fso.GetFolder(disk)
   Set fc = f.SubFolders
   For Each i in fc
      s = s & i.name
      s = s & CHR(10)
   Next
s = s + vbCr
s = s + vbCr
s =  s + "ВСЕГО ПАПОК: " & fc.Count
   WScript.Echo s
```

## ***Задание 4.***

*Прокомментируйте все строки скрипта из примера 3.3 .Запустите скрипт примера 3.3, модифицированный под анализ папок диска С.*

Наконец, рассмотрим особенности установки атрибутов файлов/папок и получением сведений о типе файла. Тип файла возвращает свойство `Type`. Вызывается оно аналогично описанным ранее свойствам:

```
WScript.Echo (File1.Type);
```

Если вызовать свойства файла по правому клику мышки, то на вкладке "Общие" в поле "Тип" будет как раз то описание, что возвращает это свойство.

Получать или устанавливать атрибуты файлам и папкам позволяет свойство `Attributes`. Так, посмотреть атрибуты нашего файла можно командой:

```
WScript.Echo (File1.Attributes);
```

В таблице 2 приведены значения, которые может принимать данное свойство.

Таблица 2. Атрибуты папок и файлов.

| Константа | Значение | Действие | Описание |
|---|---|---|---|
| Normal | 0 | чтение/запись | Обычный файл без атрибутов |
| ReadOnly | 1 | чтение/запись | Только чтение |
| Hidden | 2 | чтение/запись | Скрытый |
| System | 4 | чтение/запись | Системный |
| Volume | 8 | только чтение | Метка диска |
| Directory | 16 | только чтение | Папка |
| Archive | 32 | чтение/запись | Архивный |
| Compressed | 128 | только чтение | Сжатый файл |

Если необходимо установить несколько свойств файлу, значения надо складывать. Например, чтобы установить для файла атрибуты только для чтения, скрытый и системный, надо передать значение 1+2+4=7:

```
File1.Attributes = 7;
```

***Задание 5.***

*В пример 3.2. скрипт для создание директория и файла написан на языке JavaScript. Перепишите этот скрипт на языке VBScript. Добавьте необходимые строки, которые заполнят файл `D:\\TEST\\test.txt` фразой «Этот файл создан на языке VBScript в среде WSH».*

*Напишите скрипт на VBScript, определяющий время создания и атрибуты файла `D:\\TEST\\test.txt`.*

## Литература


1. Конспект лекций.
2. Попов А.В. Windows Script Host для Windows 2000/XP. – СПб.: БХВ – Петербург, 2004. – 640 с.: ил.