Lecture 6
Manage Folders/Files Using FSO

Print page

Welcome to our Lecture 6:)

When writing scripts for Windows Script Host, Active Server Pages, or other applications where scripting can be used, it often requires to operate folders/files, such as adding, moving, changing, creating, or deleting folders and files. In this lecture, we will learn how to use the FileSystemObject (FSO) Library to operate drives, folders and files.

Key words

FSO (FileSystemObject), FSO library, Scripting Runtime Library, WBEM (Web-Based Enterprise Management), File System, Folder Manipulation, File Manipulation

Reference to textbook chapters

This lecture covers Chapters 12 & 13. (Don Jones, VBScript, WMI and ADSI unleashed : using VBSscript, WMI, and ADSI to automate Windows administration eBook: Chapter 12. Working with the File System; Chapter 13. Putting It All Together.).

The FileSystemObject Library

FSO library contains a set of objects for variety of file operations, which allows you to create, delete, gain information about, and generally manipulate drives, folders, and files. Before we get into any details, let's practise the following script createFile.vbs.

```
Dim oFSO
Set oFSO = CreateObject("Scripting.FileSystemObject")
Set a = oFSO.CreateTextFile("c:\\testfile.txt", true)
```

```
a.WriteLine "This is a test."
a.Close
```

The code simply creates a text file and writes a message to the file. In general, to program with the `FileSystemObject` (FSO) object library, we need to consider the following three scripting steps:
✳ use the `CreateObject` method to create a `FileSystemObject` object
✳ use the appropriate method on the newly created object
✳ access the object's properties.

To create a `FileSystemObject` object, simply use the following statement.

```
CreateObject("Scripting.FileSystemObject")
```

Note that you need to use `Set` to assign the object to a variable. Once you get the object, you can use the following method to create text files, create folders, delete files, and delete folders:
✳ CreateTextFile, CreateFolder
✳ GetFile, GetFolder, GetDrive
✳ MoveFile, MoveFolder
✳ CopyFile, CopyFolder
✳ DeleteFile, DeleteFolder

See Working with Drives and Folders from Micrisoft Developer Network for the details of how to use these methods. See FileSystemObject Methods for a full list of the methods.

Operating Drives

The following code shows you how to get drive information programmatically:

```
Dim oFSO, oDrive
Set oFSO = WScript.CreateObject("Scripting.FileSystemObject")
For Each oDrive In oFSO.Drives
    IF oDrive.isReady THEN
         MsgBox "Drive " & oDrive.DriveLetter & _
         " has a capacity of " & oDrive.TotalSize & " bytes " & _
         " and is drive type " & oDrive.DriveType
    END IF
Next
```

Note that `oFSO.Drives` contains all the drive objects mounted by the operating system that is running. Drive objects represent the logical drives attached to your system, including network drives, CD-ROM drives, and so forth. Therefore they can be many.

Each drive object contains the following data members:

• AvailableSpace
• DriveLetter
• DriveType
• FileSystem
• FreeSpace
• IsReady
• Path
• RootFolder
• SerialNumber
• ShareName
• TotalSize
• VolumeName

The following code shows you how to operate a particular drive by providing a drive path `drvPath`.

```
Dim fso, d, s, drvPath
'give your drive path
drvPath = ???
Set fso = CreateObject("Scripting.FileSystemObject")
Set d = fso.GetDrive(fso.GetDriveName(drvPath))
s = "Drive " & UCase(drvPath) & " - "
s = s & d.VolumeName   &
s = s & "Free Space: " & FormatNumber(d.FreeSpace/1024, 0)
s = s & " Kbytes"
MsgBox s
```

Drives also provide an entry point into each drive's file system, starting with the root folder of the file system hierarchy. Because the `Drive` object represents one of the simplest aspects of the file system, it's one of the simplest objects in the FSO.

Operating Folders

FSO offers more methods to the Folder object for manipulating folders than the Drive object:

• CopyFolder: copies a folder.
• CreateFolder: creates a new folder.
• DeleteFolder: removes a folder permanently. Note that the deleted folder doesn't ever make it to the Recycle Bin, and there's no "Are you sure?" prompt.
• FolderExists:, like DriveExists, returns a True or False indicating whether the specified folder exists.
• GetFolder: accepts a complete folder path and, if the folder exists, returns a Folder object that represents the folder.
• GetParentFolderName: accepts a complete folder path and returns the name of its parent folder.
• GetSpecialFolder: returns the complete path to special operating system folders.
• MoveFolder: moves a file a folder.

Download the program FolderOperation.vbs to practice.

Similar to drive objects, once you call GetFolder method of FSO, you get a folder object. Each folder object have the

following methods:
- Copy
- Delete
- Move
- CreateTextFile

Using these methods is straightforward.

You can also use FSO to retrieve attribute information about folders.

Operating Files:

File operation is also similar. Download the program fileOperatrion.vbs to practice. Note that a file object contains the following properties:
- Attributes
- DateCreated
- DateLastAccessed
- DateLastModified
- Drive
- Name
- ParentFolder
- Path
- ShortName
- Size
- Type

Surprisingly, not only can you get the information about a file's or folder's attributes but also change these attributes.

```
Dim fso, f1, f2, s
Set fso = CreateObject("Scripting.FileSystemObject")
Set f = fso.GetFile("c:\test.txt")
f.attributes = 1
```

This code changes the attribute of C:\test.txt to read-only. See Attributes Property for the meaning of the attribute values.

Microsoft provides a comprehensive example code FileOperationCollection.vbs which can be run directly. It is very useful. We will practice it in the practical tasks.

At the end, I'd like to remind you again of Microsoft's Philosophy: Create a minimal core WSH structure that can interact with any number of separate object models for extention. For instance, Windows Management Instrumentation (WMI, See Lecture 5), the FileSystemObject (FSO, See Lecture 6), Active Directory Service Interfaces (ADSI, See Lecture 7), Collaboration Data Objects (CDO, See Lecture 8) object models are separate from the WSH object model (i.e. WMI, FSO, ADSI, and CDO are not built in WSH).