PREV Checkin

NEXT eme...

Recent

Topics

Tutorials

Highlights

Settings

Feedback (http://community.safa

Sign Out

Settings

10 days left in your trial. Subscribe.

Feedback
(http://community.safaribooksonline.com/)

Sign Out

Uploading a file using FTP

In a previous recipe, we have seen how to download a file using `XMLHttp`. Here we will see how to upload a file to a web server using the FTP protocol.

Getting ready

From the **File** menu, navigate to **New** | **Function Library** or use the *Alt* + *Shift* + *N* shortcut. Name the new function library `FTP.vbs`. Make sure the library is associated to the test. In order to use the code given in this recipe, you must have an FTP user account on a server. To understand this recipe, you should be familiar with FTP protocol and the command line.

How to do it...

In the function library, we will put the following code.

1. First we will define the following constants for better readability and reusability:

```
const C_FSO="Scripting.FileSystemObject"
const C_SHELL="WScript.Shell"
const C_ASCII="ascii", C_BIN="binary"
const C_FTP_CMD="%comspec% /c FTP -n -s:"
const C_SYNC_TIME=5000
const C_TEMP="%TEMP%"
const C_OPENASDEFAULT=-2
                        T_EXIST=0
                        G=1
                        G=2
                        t"
                        rompt n"
const C_PUT_OP="put "
const C_CD_OP="cd "
const C_FILE_TRANSFER_OK="226-File successfully transferred"
const C_FILE_NOT_FOUND="File not found"
const C_CANNOT_LOGIN="Cannot log in"
const C_UNKNOWN_ERROR="Unknown error"
const C_USER="USER "
const C_REDIRECT=" > "
const C_ERR_STR="Error: "
```

2. Next, we will define our `FTP` class with the following fields, which will be used to store the data needed for the FTP operation:

```
class FTP
     private m_oFSO
     private m_oScriptShell

    private m_sLocalFile
    private m_sPassword
    private m_sRemotePath
    private m_sResultsTmpFilename
    private m_sScriptTmpFilename
    private m_sSite
    private m_sTType
    private m_sUsername
```

3. We will then initialize our class upon instantiation with a `FileSystemObject` to handle files and a `WScript.Shell` object to handle commands:

```
private sub class_initialize
       FSO=CreateObject(C_FSO)
         ScriptShell=CreateObject(C_SHELL)
         'default transfer type - binary
         TType=C_BIN
    end sub
```

4. We will take care of disposing of the same objects at the time the object is destroyed:

```
private sub class_terminate
```

Enjoy Safari? Subscribe Today

```
        FSO=nothing
          ScriptShell=nothing
    end sub
```

5. Then we will initialize the object with our FTP account connection and login information:

```
function init(sSite, sUsername, sPassword)
        Site=sSite
        Username=C_USER & sUsername
        Password=sPassword
    end function
```

6. Next, upload the local file with the following method:

```
function uploadFile(sLocalFile, sRemotePath, sTType)
        LocalFile=C_PUT_OP & sLocalFile
         RemotePath=C_CD_OP & sRemotePath
         TType=sTType

        me.createFTPScript()
'Run the FTP command through the command line
'we use the WScript.Shell object to run the FTP command
'through the command line, in which the -n switch
'suppresses auto-login upon initial connection and the -s:
'switch takes the path of the temporary script file we
'created as parameter. The commands contained in such a
'file run automatically after FTP starts. The > operator
'redirects the FTP verbose output to our temporary results
'file. The last parameter is set to TRUE and it indicates
'whether to wait until the FTP program finishes.
ScriptShell.Run C_FTP_CMD & ScriptTmpFilename & " " & Site & C_RED

        Wait 0, C_SYNC_TIME

        uploadFile=me.checkResults()

        if FSO.FileExists(ScriptTmpFilename) then
             FSO.DeleteFile(ScriptTmpFilename)
        end if
    end function
```

7. Check the results of the transfer, which are stored in `ResultsTmpFile` with the following method:

```
function checkResults()
        dim fFTPResults, sResults

        'Check results of transfer.
        Set fFTPResults = FSO.OpenTextFile(ResultsTmpFilename, C_F
        if not fFTPResults.AtEndOfStream then
            sResults = fFTPResults.ReadAll
        end if
        fFTPResults.Close

        if FSO.FileExists(ResultsTmpFilename) then
            FSO.DeleteFile(ResultsTmpFilename)
        end if

        If InStr(sResults, C_FILE_TRANSFER_OK) > 0 Then
          checkResults = micPass
        ElseIf InStr(sResults, C_FILE_NOT_FOUND) > 0 Then
          checkResults = micFail
          sResults=C_ERR_STR&C_FILE_NOT_FOUND&vbNewLine&sResults
        ElseIf InStr(sResults, C_CANNOT_LOGIN) > 0 Then
          checkResults = micFail
          sResults=C_ERR_STR&C_CANNOT_LOGIN&vbNewLine&sResults
        Else
          checkResults = micFail '"Error: Unknown."
          sResults=C_ERR_STR&C_UNKNOWN_ERROR&vbNewLine&sResults
        End If

        reporter.ReportEvent checkResults, typename(me) & ".uploadF
    end function
```

8. The `CreateFTPScript()` function creates the FTP script from the input file `fFTPScript`:

```
function createFTPScript()
        dim fFTPScript        'As file
        dim sFTPScript        'As string
        dim sFTPTempPath    'As string
        dim sFTPTempFile    'As string

        'Input file for ftp command
        sFTPScript=join(array(Username, Password, RemotePath, TTyp

        sFTPTempPath = ScriptShell.ExpandEnvironmentStrings(C_TEMP

        ScriptTmpFilename = sFTPTempPath & "\" & FSO.GetTempName
        ResultsTmpFilename = sFTPTempPath & "\" & FSO.GetTempName

        'Write the input file for the ftp command to a temporary f
        Set fFTPScript = FSO.CreateTextFile(ScriptTmpFilename, Tru
        fFTPScript.WriteLine(sFTPScript)
        fFTPScript.Close
        Set fFTPScript = Nothing
    end function
```

9. Add the following properties as accessors to the fields:

```
public property get FSO()
        set FSO=m_oFSO
    end property
    public property let FSO(oFSO)
        set m_oFSO=oFSO
    end property
```

```
    public property get LocalFile()
        LocalFile=m_sLocalFile
    end property
    public property let LocalFile(sLocalFile)
        m_sLocalFile=sLocalFile
    end property

    public property get Password()
        Password=m_sPassword
    end property
    public property let Password(sPassword)
        m_sPassword=sPassword
    end property

    public property get RemotePath()
        RemotePath=m_sRemotePath
    end property
    public property let RemotePath(sRemotePath)
        m_sRemotePath=sRemotePath
    end property

    public property get ResultsTmpFilename()
        ResultsTmpFilename=m_sResultsTmpFilename
    end property
    public property let ResultsTmpFilename(sResultsTmpFilename)
        m_sResultsTmpFilename=sResultsTmpFilename
    end property

    public property get ScriptTmpFilename()
        ScriptTmpFilename=m_sScriptTmpFilename
    end property
    public property let ScriptTmpFilename(sScriptTmpFilename)
        m_sScriptTmpFilename=sScriptTmpFilename
    end property

    public property get ScriptShell()
        set ScriptShell=m_oScriptShell
    end property
    public property let ScriptShell(oScriptShell)
        set m_oScriptShell=oScriptShell
    end property

    public property get Site()
        Site=m_sSite
    end property
    public property let Site(sSite)
        m_sSite=sSite
    end property

    public property get TType()
        TType=m_sTType
    end property
    public property let TType(sTType)
        select case lcase(sTType)
            case C_ASCII, C_BIN
               m_sTType=lcase(sTType)
            case else
               m_sTType=C_BIN
        end select
    end property

    public property get Username()
        Username=m_sUsername
    end property
    public property let Username(sUsername)
        m_sUsername=sUsername
    end property
end class
```

The GetFTP method is used as a constructor for the FTP object:

```
function getFTP(sSite, sUsername, sPassword)
    dim oFTP
    on error resume next
    set oFTP=new FTP
     call oFTP.init(sSite, sUsername, sPassword)
     if err.number<>0 then set oFTP=nothing
     set getFTP=oFTP
end function
```

10. Finally, in `Action1`, we will invoke the FTP upload function with the following code, which creates an FTP custom object (based on our class) and uploads the file.

```
dim sSite, sUsername, sPassword, sLocalFile, sRemotePath

sSite="www.mysite.com"
sUsername="admin"
sPassword="mypassword"
sLocalFile="mylocalpathname\" & "mylocalfile.txt"
sRemotePath="//ftp/admin/"

set oFTP=getFTP(sSite, sUsername, sPassword)
if not oFTP is nothing then
    call oFTP.uploadFile(sLocalFile, sRemotePath, "")
    set OFTP=nothing
else
    reporter.ReportEvent micFail, "FTP.uploadFile", "Could not cre
end if
```

How it works...

First, we define the five variables required by the FTP protocol:

- `strSite`: This is the URL of our FTP server
- `strUsername`: This is the name of our FTP account

- `strPassword`: This is the password for our FTP account
- `strLocalFile`: This is the file to be uploaded
- `strRemotePath`: This is the path on our FTP account in which we put our file

We then retrieve an instance of our `FTP` class by calling the `getFTP` method with three arguments, namely, `strSite`, `strUsername`, and `strPassword`. A check is performed to verify that a valid object was returned, and then the `uploadFile` method is called with two arguments, `strLocalFile` and `strRemotePath`.

In the `uploadFile` method, we use these arguments to build a string with the FTP commands required to perform the upload operation. The transfer type is validated (ASCII or binary) in the `TType` property, with binary as the default type. A call to `createFTPScript` writes this string to a temporary file, which then serves as input script for the FTP command. We also create a temporary filename to store the results of the upload operation.

Next, we use the `WScript.Shell` object to run the FTP command through the command line, in which the `-n` switch suppresses auto login upon initial connection, and the `-s` switch takes the path of the temporary script file that we created as parameter. The commands contained in such a file run automatically after FTP starts. The `>` operator redirects the FTP verbose output to our temporary results file. The last parameter is set to `TRUE`, and it indicates whether to wait until the FTP program finishes or not. After it does, we call `checkResults()` to read the contents of the output file and return whether our upload was successful or not according to the status returned by FTP, which is written to the file.

PREV
Checking whether…
NEXT
Identifying eleme…
Recommended / Queue / Recent / T
y.safaribooksonline.com/) /
Sign Out
© 2015 Safari.
Terms of Service / Privacy Policy