Website: www.chestysoft.com Email: info@chestysoft.com

csASPUpload Version 1.2 - ASP File Upload Component from Chestysoft

Introduction

This component is used to save files that have been attached to HTML forms using the "<input type=file…>" tag. It can be used to save the file or files to disk on the server or it can be used to save to a binary field in a database. The form attributes "method=post" and "enctype=multipart/form-data" must be used.

Any number of files can be attached and form variables can be passed as usual. However, the Request.Form collection cannot be used and alternative properties for reading form variables are provided with this component and are described in the section "Form Variable Properties".

The ActiveX file csASPUpload.dll should be registered on the server using REGSVR32.EXE. Regsvr32 is usually to be found in the Windows System folder, and it runs at the command prompt using the syntax:

regsvr32 *dllname*

where *dllname* is the path and name of the dll to register. Chestysoft has a free utility that performs this function through a Windows interface. For more information on this, visit the Chestysoft web site, or follow: www.chestysoft.com/dllregsvr/default.asp

Object Creation

The code:

```
Set Upload = Server.CreateObject("csASPUpload.Process")
```

- creates an instance of the object called “Upload”.

In the trial version, use:

```
Set Upload = Server.CreateObject("csASPUploadTrial.Process")
```

Visit the Chestysoft web site for details of how to buy the full version - www.chestysoft.com

After creating the object the component properties are set and can be read and any uploaded files can be saved.

Form Variable Properties

The form variables cannot be read using the Request.Form collection, so properties are included specifically to access the form variables**.** Do not use any Request.Form commands on the same ASP page as the csASPUpload component as an error will be generated.

VarCount - the number of variables received.

Value(“*VariableName*”) - the value of a named variable, *VaiableName*

Example:

```
Response.Write Upload.Value("UserName")
```

- displays the value of the form variable “UserName”.

VarByIndex(*index*) - the variables are stored in a zero based array and can be retrieved using the index.

Example:

```
Upload.VarByIndex(0)
```
- returns the value of the first variable.

VarName(*index*) - returns the name of the variable given its index.

Example:

```
Upload.VarName(0)
```
- returns the name of the first variable.

File Properties

File properties are also stored as zero based arrays.

FileQty - Returns the number of files received. This property is important as a check that files have been received, and for looping when multiple files are uploaded.

Example:

```
If Upload.FileQty = 0 Then
    Response.Write "No files received"
End If
```

FileName(*index*) - the name of the file complete with extension.

LocalName(*index*) - the full path complete with name and extension, as the file was stored on the client computer.

Extension(*index*) - the file extension without the period character.

NameOnly(*index*) - the file name without the extension.

NewName(*index*) - if the file is renamed during saving to prevent overwriting, this property is set to the new file name. See also the section on OverwriteMode.

ContentType(*index*) - the content encoding type e.g. “image/gif”

FileSize(*index*) - the size of the file in bytes.

Example:

```
<p>The file size is <%= Upload.FileSize(0) %></p>
```

This would display the size of the first file in the array.

FileData - the file as binary data. This is used when saving to a database. It could also be used with the Response.BinaryWrite method to stream the file back to the browser.

Example:

```
RSet("Image") = Upload.FileData(0)
```

This line would write the first file in the array to the database field “Image”, assuming a recordset called “RSet” had been opened. In MS Access, the data type for the field would be ”OLE Object”.

Version - The version of the csASPUpload.dll file.

Note that when the form allows only one file to be uploaded, the index will always be zero. If the form is submitted with no file attached, i.e. the input field is blank no file will be added to the FileQty total. Use FileQty to loop through arrays and to prevent index errors. Use FileSize to check that a file actually contains data.

The FileSave Method

The syntax is: **FileSave *filename , index***

Example:

```
Upload.FileSave "C:\temp\newfile.gif", 0
```

This saves the first file in the array using the path and name stated. Note that the index is after the comma and no brackets are used.

Note: For any ASP script to save a file on the server, the Internet User Guest Account must have write permissions for that particular directory.

Overwrite Mode

The property **OverwriteMode** has the value 0, 1 or 2 and it determines what happens if the FileSave method attempts to write to a file that already exists. The default value is 0.

OverwriteMode = 0	-	Any existing file will be overwritten.
OverwriteMode = 1	-	The new file will not be saved if an existing file has the same name.
OverwriteMode = 2	-	If the new file name matches an exis ting name the characters "~x" will be added at the end where "x" is the smallest number needed to make the name unique.

When an OverwriteMode of 2 is used, the file property **NewName** will be set to the name of the file that was actually saved.

Example:

```
Upload.OverwriteMode = 2
Upload.FileSave "C:\temp\newfile.gif" , 0
Response.Write Upload.NewName(0)
```

This will save the uploaded file as "newfile.gif", if that name is not already used. If a file with that name exists, it will save it as "newfile~1.gif" (or newfile~2.gif … etc.) The name actually used will be written out in the Response.Write statement.

Simple Examples of Saving Files

Here are some simple examples of using the *FileSave* method to save files on the server. In each case the files are saved in the same directory as the script. Replace *Upload.CurrentDir* with the directory path to save somewhere else. Each example uses the FileQty property to check for at least one file before proceeding. This prevents array index errors.

Example 1 - Single file

```
<%
Set Upload = Server.CreateObject("csASPUpload.Process")
If Upload.FileQty > 0 Then
  Upload.FileSave Upload.CurrentDir & Upload.FileName(0), 0
  Response.Write "<p>File saved</p>"
Else
  Response.Write "<p>No file received</p>"
End If
%>
```

Example 2 - Single file where uploads bigger than 1 MB are rejected.

```
<%
Set Upload = Server.CreateObject("csASPUpload.Process")
If Request.TotalBytes < 1048576 Then   '(no of bytes in 1 MB)
  If Upload.FileQty > 0 Then
    Upload.FileSave Upload.CurrentDir & Upload.FileName(0), 0
    Response.Write "<p>File saved</p>"
  Else
    Response.Write "<p>No file received</p>"
  End If
Else
  Response.Write "<p>File too big</p>"
End If
%>
```

Example 3 - Multiple uploads using *OverwriteMode* to rename duplicates

```
<%
Set Upload = Server.CreateObject("csASPUpload.Process")
If Upload.FileQty > 0 Then
  Upload.OverwriteMode = 2
  For Index = 0 to Upload.FileQty - 1
    Upload.FileSave Upload.CurrentDir & Upload.FileName(Index), Index
  Next
  Response.Write "<p>Files saved</p>"
Else
  Response.Write "<p>No file received</p>"
End If
%>
```

Obsolete Properties and Methods

Before version 1.2 there was a property called "MaxKB" and a method called "Read". *MaxKB* was used to reject uploads bigger than a specified size. This has been removed. Use *Request.TotalBytes* to check the upload size. *Read* has also been removed. Calling either of these will not produce an error.

File Utilities

There are a number of file utility functions included for convenience. They are not intended to be a comprehensive set, because asp has the built in File Access component to cover most file utilities. These are the functions most likely to be useful while processing file uploads.

FileExists(*FileName*) - Returns a Boolean value. *FileName* is the physical path and file name of the file in question.

CurrentDir - Returns the physical path of the directory containing the script. It is complete with the trailing backslash character.

ParentDir(*Directory*) - *Directory* is a string value and must be a full physical directory path. The return value is the parent directory.

Example:

```
Response.Write Upload.ParentDir(Upload.CurrentDir)
```

This would display the parent directory to the one containing the current script.

Delete(*FileName*) - This deletes the file *FileName*. Note that it is permanently deleted, NOT placed in the Recycle Bin.

Copy *OldName, NewName* - This copies the file *OldName* to the location and name given by *NewName*. Again, full paths are required.

Rename *OldName, NewName* - This renames the file *OldName* to *NewName*. Full paths are required, and so renaming to a different directory is the equivalent of moving the file.

AppendToFile *FileName, NewLine* - This appends the string *NewLine* to the text file *FileName*. If the text file does not exist, it will be created if possible. The full server path is required.

Example:

```
Upload.AppendToFile Upload.CurrentDir & "test.txt", "Hello"
```

This will append the line "Hello" at the end of a text file called test.txt which is in the same directory as the current script. If the file does not exist it will create it.

AppendToFile is the only command in this component for manipulating text files. It is useful for maintaining a simple log or to assist with testing. There is a full set of commands for dealing with text files in the built in File Access component.

All the file handling routines require that the Internet Guest Account has the appropriate permissions on the server, otherwise errors will result.

The Database Example

This example is supplied in the download with the component and it shows the use of csASPUpload by saving images to an Access database. In this case it stores the images as raw data within the database. The csASPUpload.dll component must have been registered on the server and the following files must be copied into a directory that is web shared, with permission to run scripts:

dbview.asp, dbview1.asp, delete.asp, imageview.asp, process.asp, upload.htm, ,ImageDB.mdb & access.dsn

The page which uses the csASPUpload component is “process.asp”. Note that the database connection is made using the following connection string, which is assigned to the variable "DataConnection":

```
DataConnection = "PROVIDER=MICROSOFT.JET.OLEDB.4.0;DATA SOURCE=" & _
   DirPath & "ImageDB.mdb"
```

This uses OLE DB for opening the connection, which is the preferred method. The previous line is remarked and shows the alternative using the file DSN. The files "dbview.asp", "dbview1.asp", "delete.asp" and "process.asp" all make a connection to the database, and so any changes made to one must be applied to the others.

The file “Upload.htm” is the starting page.

Revision History

The current version of csASPUpload is 1.2

<u>New in Version 1.1:</u>

File utilities included for copying, renaming and deleting files and for obtaining the physical path on the server.
OverwriteMode introduced.

<u>New in Version 1.2:</u>

Maximum file size removed.
MaxKB property and Read method obsolete.
User friendly error messages added for array index errors.

Other Components From Chestysoft

Visit the Chestysoft web site for details of other asp components.

csImageFile	-	Resize and join image. Ideal for use with csASPUpload.
csFileDownload	-	Control file downloads with an asp script.
csDrawGraph	-	Draw pie charts and bar charts with an asp script.
csWAPDraw	-	Create and edit wireless bitmaps for WAP devices.
csIniFile	-	Use Windows style inifiles in your asp applications.

Chestysoft, October 2002.
www.chestysoft.com