Website: www.aspphotoresizer.com

Email: info@aspphotoresizer.com

ASPFileSaver - File Upload Component

This is an ASP component for saving files that have been uploaded through a browser using the <input type="file"> tag. It supports uploads of single files and can either save this file or export it as a variant array for storing in a binary database field or exporting to our ASPPhotoResizer component. In ASP the Request.Form collection cannot be used when files are uploaded and so properties are provided for reading form variables.

A free, fully functional trial version of ASPFileSaver is available. If you are reading this instruction manual for the first time, it is likely that you have just downloaded the trial version. The trial version has a built in expiry date that causes it to stop working after that time. This is the only difference in functionality between the trial and full versions. This means that you can fully test if this control is suitable for your application before considering whether to license the full version.

Using these Instructions

These instructions are divided into a number of sections covering different types of functions available in ASPFileSaver. A full Table of Contents is available on the next page and an index listing all commands in alphabetical order is included at the back for easy reference.

Click on one of the links below to go directly to the section of interest:

ASPPhotoResizer.com, January 2010.
www.aspphotoresizer.com

TABLE OF CONTENTS

1. Registering the Component and Getting Started

1.1. Registration and Server Permissions

Before the component can be used the DLL file, ASPFileSaver.dll (or ASPFileSaverTrial.dll for the trial version) must be registered on the server. This can be done using the command line tool REGSVR32.EXE which should be in the Windows System folder. The syntax is:

regsvr32 *dllname*

where *dllname* is the path and name of the DLL to register. We can recommend a free utility that performs this function through a Windows interface which can be easier although the result is identical. This tool can be downloaded here: www.chestysoft.com/dllregsvr/default.asp

The application that uses the component must have permission to read and execute the DLL. In a web application like ASP this means giving the Internet Guest User account Read and Execute permission on the file. This account must also have the appropriate permissions for any file reading, writing or deleting that the component is to perform.

1.2. Object Creation

In any script or programme that uses the component an object instance must be created. The syntax in ASP is as follows.

For the full version:

```
Set Upload = Server.CreateObject("ASPFileSaver.FileSave")
```

For the trial version:

```
Set Upload = Server.CreateObject("ASPFileSaverTrial.FileSave")
```

In both cases the object name is "Upload", but any variable name could be used.

1.3. The Trial Version

The trial version of the component is supplied as a separate DLL called ASPFileSaverTrial.dll, with a class name of "ASPFileSaverTrial.FileSave". This trial version is fully functional but it has an expiry date, after which time it will stop working. After the expiry date an error message will be shown on creation of the object.

The expiry date can be found by reading the *Version* property.

Version - String, read only. This returns the version information and for the trial, the expiry date.

Example:

```
Set Upload = Server.CreateObject("ASPFileSaverTrial.FileSave")
Response.Write Upload.Version
```

Visit our web site to buy the full version - http://www.aspphotoresizer.com

1.4. The Upload Form

The HTML page that posts a file upload must contain a form that has the method and enctype attributes set as follows:

```
<form action="filesave.asp" method="post" enctype="multipart/form-data">
```

The action attribute must contain the address of the ASP script that will use the ASPFileSaver component to save the file. The method must be "post" and the enctype must be "multipart/form-data".

The file is selected using the input using the following form tag:

```
<input type="file" name="filesent">
```

This will display a text field and "Browse" button where the user can enter or select the local file on their system. Note that this field cannot be filled using Javascript. ASPFileSaver will only save a single uploaded file so if multiple Browse buttons are used, it will always be the first file that is saved. The name attribute is not used by ASPFileSaver but it must be present in the input tag.

2. Saving or Exporting the Uploaded File

The uploaded file can be saved to disk on the server using *SaveFile*. It can be exported as a variant array using *FileAsVariant*, and this would be used to send an image file to the ASPPhotoResizer component for processing or if the file was to be stored in a binary database field. Some properties are available to provide information about the file, such as *FileSize* and *Filename*.

2.1. Saving to Disk

SaveFile *Filename* - This saves the uploaded file to the full physical path specified in *Filename*. The Internet Guest User must have Write permission on the destination directory to save the file and Modify permission to overwrite an existing file.

Example using a hard coded file name:

```
Upload.SaveFile "C:\files\newfile.jpg"
```

Use Server.MapPath to convert a virtual path to a physical path.

2.2. Exporting As Binary Data

FileAsVariant - Variant array, read only property. The uploaded file exported as a binary variable.

Example of saving the file to a binary database field called "Image":

```
RSet("Image") = Upload.FileAsVariant
```

Example of exporting the file to the ASPPhotoResizer component for further processing:

```
JPG.VariantIn = Upload.FileAsVariant
```

This assumes the instance of ASPPhotoResizer is called "JPG".

2.3. File Properties

The following properties provide information about the uploaded file.

ContentType - String. The MIME type specified during the upload.

Filename - String. The name of the file that was uploaded, complete with extension.

FileExtension - String. The file extension, without the period character.

FilenameNoExtension - String. The file name, with the extension removed.

FileSize - Integer. The size of the uploaded file in bytes. This property can be used to check if a file was uploaded, because it will always be zero when no file is uploaded.

NewName - String, read only. 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*.

2.4. OverwriteMode

The *OverwriteMode* property provides a quick way to prevent files with duplicate names from being overwritten.

OverwriteMode - Integer, 0, 1 or 2. This property determines what happens if the SaveFile 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 existing 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.SaveFile "C:\temp\newfile.gif"
Response.Write Upload.NewName
```

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.

If an alternative character to the '~' is needed this can be assigned to the property *OverwriteChr*. This might be necessary on Windows 2003 or if URLScan is used because URLs containing this character may be blocked from downloading.

OverwriteChr - String. The character or characters added before the number when a file is renamed using *OverwriteMode*. (Default = "~")

2.5. Some Examples

Example 1 - Saving a file with no error checking.

```
<%
  Set Upload = Server.CreateObject("ASPFileSaver.FileSave")
  Upload.SaveFile Server.MapPath(".") & "\" & Upload.FileName
%>
```

This will save the file into the same directory as the script and it will use the existing file name for the saved file. If a file already exists by that name it will be overwritten. An error will be generated if the user submits the form without attaching a file.

Example 2 - Checking for an upload before saving.

```
<%
  Set Upload = Server.CreateObject("ASPFileSaver.FileSave")
  If Upload.Filesize > 0 Then
    Upload.SaveFile Server.MapPath(".") & "\" & Upload.FileName
    Response.Write "<p>File saved.</p>"
```

```
  Else
    Response.Write "<p>No file submitted.</p>"
  End If
%>
```

This checks the Filesize property, which will be zero if no file was uploaded. The If..Then..Else statement will save the file if there is a file.

Example 3 - Using a form variable for the saved file name.

In this example a form variable is added to the form for the user to specify the name that will be used when the file is saved.

```
<input type="text" name="SaveName">
```

The script is similar to example 2 above except the variable name is used for the saved name and the file is saved into a sub directory. The previous examples have saved the files into the same directory as the script but a separate location would often be preferable.

```
<%
  Set Upload = Server.CreateObject("ASPFileSaver.FileSave")
  If Upload.Filesize > 0 Then
    Upload.SaveFile Server.MapPath(".") & "\files\" &_
Upload.FormVariable("SaveName") & "." & Upload.FileExtension
    Response.Write "<p>File saved.</p>"
  Else
    Response.Write "<p>No file submitted.</p>"
  End If
%>
```

In practice, the file name would rarely be chosen like this and some error checking would be needed to make sure the name was valid. This example shows how to read a form variable and construct a name based on a variable and the existing extension.

The functions for reading form variables are described in the next section. Request.Form cannot be used with file uploads. Request.Querystring can be used as normal.

3. Form Variables

Form variables cannot be read using Requst.Form, so the following properties must be used instead. *FormVariable* reads the variable given its name. *FormVariableCount* returns the number of variables in the form, and *FormVariableByIndex* returns the value given the index of the variable within the form.

FormVariable (*VariableName*) - String. The value of the variable where *VariableName* is the string name of the variable.

Example:

```
Response.Write Upload.FormVariable("User")
```

This will display the value for a form variable called "User".

FormVariableCount - Integer. The number of form variables in the form. This includes the name of the uploaded file.

FormVariableByIndex (*Index*) - String. The value of the variable specified by *Index*. *Index* represents the number of the variable within the form, where the first has an index of zero and the last has an index of *FormVariableCount* - 1.

Example of looping through all the form variables:

```
For I = 0 to Upload.FormVariableCount - 1
  Response.Write Upload.FormVariableByIndex(I) & "<br>"
Next
```

4. Troubleshooting

This section covers some of the common causes for errors when using the ASPFileSaver component.

4.1. Failure to Register the Component

The component must be registered on the server as described in Section 1.1. If the component is not registered it will generate the error:

Server object, ASP 0177 (0x800401F3)
Invalid ProgID.

4.2. Incorrect Class Name

If the class name inside the Server.CreateObject command is incorrect it will generate exactly the same error as when the component is not registered. Take care to spell the class name correctly. Also, note that the class name is different for the trial version, compared with the full version. See Section 1.2 for more on the class names and object creation.

4.3. Insufficient Permission on the DLL

An ASP script runs under the Internet Guest User account, unless specifically configured otherwise. This account must have Read and Execute permission on the DLL file, ASPFileSaver.dll. Failure to do this will result in the error:

Server object, ASP 0178 (0x80070005)
The call to Server.CreateObject failed while checking permissions. Access is denied to this object.

4.4. Insufficient Permission to Save Files

The Internet Guest User account must have permission to write into the directory where the files are to be saved. Insufficient permissions will result in the following error:

ASPFileSaver.FileSave (0x8000FFFF)
Error saving file. Check the user permissions.

Modify permission is required to overwrite existing files.

4.5. Using Request.Form in the Script

Request.Form commands must not be used in the same script as the ASPFileSaver component. The exact error generated will depend on where the Request.Form command appears in the script.

Section 3 describes how to read form variables.

4.6. Windows 2003 Server - ASPMaxRequestEntityAllowed

Windows 2003 (IIS 6) contains a property that limits the amount of data that can be received using a POST operation, and this defaults to the relatively low value of 200 KB. This property is called *ASPMaxRequestEntityAllowed* and it can be changed to a higher value by editing the metabase.xml file.

5. Alphabetical List of Commands