ASPPhotoResizer
JPG Resize Component

Website: www.aspphotoresizer.com

Email: info@aspphotoresizer.com

ASPPhotoResizer - Instructions

This is an ASP component / COM for resizing JPG images. Images can be resized server side "on the fly" and saved to disk or streamed to the browser. JPG compression can be changed, as can the DPI of the image. Functions are provided for crop, flip and rotate.

A free, fully functional trial version of ASPPhotoResizer 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 ASPPhotoResizer. 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:

- Registering the Component and Getting Started
- Full Table of Contents
- Alphabetical Index of Commands
- Import and Export of JPG Images
- Image Resizing
- Crop, Flip and Rotate
- Language Specific Issues (ASP, Cold Fusion)

ASPPhotoResizer.com, June 2006
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, ASPPhotoResizer.dll (or ASPPhotoResizerTrial.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. For network access the ASPPhotoResizer component must be added to a COM+ Application in Component Services.

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 JPG = Server.CreateObject("ASPPhotoResizer.Resize")
```

For the trial version:

```
Set JPG = Server.CreateObject("ASPPhotoResizerTrial.Resize")
```

In both cases the object name is "JPG", 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 ASPPhotoResizerTrial.dll, with a class name of "ASPPhotoResizerTrial.Resize". This trial version is fully functional but it has an expiry date, after which time it will stop working. The object can still be created after the expiry date but it cannot load or create images.

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 JPG = Server.CreateObject("ASPPhotoResizerTrial.Resize")
Response.Write JPG.Version
```

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

2. Import and Export of JPG Images

Images can be read into the control and exported from the control in two different ways, either as files on disk or as variant array variables.

Only images in JPG format can be imported or exported.

2.1. Loading and Saving Files From/To Disk

JPG images are loaded into the component using *LoadFromFile* and saved to disk using *SaveToFile*.

Appropriate permissions must be set for reading and writing files. In a web application the Internet Guest User account must have Read permission to open a file, Write permission to save and Modify to overwrite an existing file. The ASPPhotoResizer component must be added to a COM+ Application in Component Services before it can be used to access files across a network.

LoadFromFile *FileName* - Reads a JPG image from disk. *FileName* must be a complete physical path to the file, including the file extension.

Example:

```
JPG.LoadFromFile "C:\images\test.jpg"
```

In ASP a virtual path can be mapped to a physical path using Server.MapPath:

```
JPG.LoadFromFile Server.MapPath(".") & "\test.jpg"
```

SaveToFile *FileName* - Saves the currently loaded image to disk. *FileName* must be a complete physical path to the file, including the file extension.

2.2. Importing and Exporting Files From/To a Variant Array

JPG images can be read from a variant array variable using *VariantIn*. The main uses for this are to read from an upload component such as our ASPFileSaver component, or to read from a binary database field. Images can be exported to a variant array variable using *VariantOut*, either to stream an image to a browser or to save to a binary database field.

The *StreamImage* function will set the Content Type to "image/jpeg" and stream the current image using BinaryWrite in a single command.

VariantIn *Data* - This reads a JPG into the component from a variant array variable, *Data*. This could be a VBScript variable, a binary database filed or from our ASPFileSaver component.

To read a file directly from ASPFileSaver using ASP:

```
JPGObj.VariantIn UploadObj.FileAsVariant
```

This will read a JPG into ASPPhotoResizer from ASPFileSaver. JPGObj is the name of the instance of ASPPhotoResizer and UploadObj is the name of the instance of ASPFileSaver.

To read data from a database field in ASP the data must first be passed to a temporary variable:

```
TempData = RSet("JPGImage")
JPGObj.VariantIn TempData
```

This would read in a JPG file from a binary field in a database, assuming the field name is "JPGImage" and the recordset RSet has been opened. The temporary variable is needed to convert the data to the correct format.

VariantOut - The JPG as a variant array.

Example of saving the image to a binary database field:

```
RSet("Image") = JPGObj.VariantOut
```

VariantOut can be used to send the JPG to a browser an ASP script would use the following, without any other output appearing in the script:

```
Response.ContentType = "Image/jpeg"
Response.BinaryWrite JPGObj.VariantOut
```

StreamImage - A single command to stream the current image to a browser. This sets the response buffer to true, sets the expiry time of the page to zero to prevent caching and sets the content type to "image/jpeg". Finally it sends the data using the ASP BinaryWrite command. No other output must be used on the page.

Example of loading an image, scaling to 25% and streaming to the browser:

```
<%
Set JPG = Server.CreateObject("ASPPhotoResizer.Resize")
JPG.LoadFromFile "C:\images\big.jpg"
JPG.ScaleImage 25
JPG.StreamImage
%>
```

3. Resizing Images

The image loaded into the component can be resized in three different ways. The *ResizeImage* function specifies a new height and / or width. *ScaleImage* specifies a scale factor. *ResizeBox* will resize the image to fit a rectangular area given the height and width, maintaining the aspect ratio.

ResizeImage *Width, Height* - Resizes the current image to new dimensions *Width* x *Height*. If either parameter is zero the other will be used to determine the new size and the aspect ratio will be unchanged.

Example:

```
JPG.ResizeImage 200, 0
```

This will make the new image 200 pixels wide and the aspect ratio will be unchanged. The height will be adjusted to suit.

ScaleImage *Factor* - Scales the current image by a percent scale factor of *Factor*.

Example:

JPG.ScaleImage 50

This will scale the current image to make it half size. That would make a 300 x 200 image into a 150 x 100 image.

ResizeBox *Width*, *Height* - This will reduce the size of the current image if it is wider than *Width* or higher than *Height*. The new size will be the biggest size possible that will fit inside the area defined by *Width* x *Height* and the aspect ratio will be maintained. *ResizeBox* will never increase the size of an image.

4. Image Properties

ASPPhotoResizer can be used to change the amount of JPEG compression by changing the *JPGQuality* property before exporting an image. The pixel density can be read or changed using the *DPI* property. The current width and height can be found by reading the *Width* and *Height* properties.

Most JPG images are saved in 24 bit RGB colour but they can be saved as 8 bit greyscale. Reading a greyscale image will set the *Grayscale* property and setting the *Grayscale* property before saving will cause images to be exported as greyscale.

JPG images can also be saved using CMYK colour space. ASPPhotoResizer can read these but they will converted to RGB. There is no functionality to convert from RGB to CMYK.

JPGQuality - Integer. This is a value between 1 and 100 measuring the quality of the image when it is saved or exported. The higher the value the higher the quality and the less the amount of compression used. The default value is 90. This value is not set when a new image is read.

Example of setting *JPGQuality* to 70:

```
JPG.JPGQuality = 70
```

DPI - Integer. The pixel density of the current image in dots per inch. This property is set when an image is read, and it can be edited.

Width - Integer, read only. The width of the current image in pixels.

Height - Integer, read only. The height of the current image in pixels.

The *Height* and *Width* properties cannot be changed directly. Use one of the resize or scale functions to change the image dimensions.

Grayscale - Boolean. This is true when the current image is in 8 bit greyscale format. It is set when an image is loaded and it can be edited. There is a duplicate property called *Greyscale* so either spelling will be accepted.

5. Image Manipulation (Rotate, Crop and Flip)

This section describes functions for manipulating the image.

RotateImage *Angle* - The current image is rotated by *Angle* degrees anticlockwise, where *Angle* is a real number (single precision). Rotations by an angle that is not a multiple of 90 degrees create triangular areas on the image that are the colour specified by *BackgroundColor*.

CropImage *X1, Y1, X2, Y2* - Crops the current image to a rectangle with opposite corners at (X1, Y1) and (X2, Y2). These co-ordinates are measured from the top left of the image. If the cropping rectangle is larger than the original the image will increase in size and the new area will be the colour of *BackgroundColor*.

BackgroundColor - OLE_COLOR property. This is the colour used to fill new areas created by the *RotateImage* or *CropImage* commands.

FlipHorizontal - Flips (reflects) the image around a horizontal axis so that the top becomes the bottom and the bottom becomes the top.

FlipVertical - Flips (reflects) the image around a vertical axis so that the left becomes the right and the right becomes the left.

6. Streaming an Image to the Browser

An active server page will return HTML output by default. An HTML page is formatted text which can include spaces to display images. The images themselves are not part of the HTML but are separate files, the location of which is specified inside the IMG tag. An ASP page can be an image if the ContentType is set to "image/jpeg" and the binary data of the image is output using the Response.BinaryWrite command. The ASP image is generated by placing the path to the script inside the IMG tag.

For example, this page will display the image produced by "resize.asp":

```
<html>
<head><title>HTML page containing an image</title></head>
<body>
<img src="resize.asp">
</body>
</html>
```

Resize.asp may look like this:

```
<%@ language=vbscript %>
<%
      Set JPG = Server.CreateObject("ASPPhotoResizer.Resize")
      JPG.LoadFromFile "C:\images\big.jpg"
      JPG.ScaleImage 25
      JPG.StreamImage
%>
```

When the first HTML page is loaded it looks for the image at "resize.asp", runs the script and is sent a stream of binary data in JPG format, so the browser displays the image. It is not possible to place the StreamImage command inside the IMG tag to produce the image, it must be in a separate file.

It is useful to know that parameters can be passed to the ASP image script using the URL string and this can be read using Request.QueryString and used somewhere in the script.

Example:

```
<img src="image.asp?NewSize=150">
```

7. Language Specific Issues

All the examples that are shown with the command descriptions use ASP and VBScript. The ASPPhotoResizer component is a COM object and can be used in most COM enabled environments running on a Windows platform. We concentrate on ASP in these instructions because it is the most commonly used environment, but in this section we also show the syntax for Cold Fusion.

7.1. Active Server Pages

ASP and VBScript is already covered in these instructions but the following points are worth noting.

Calls to methods (functions) do not use brackets. For example:

```
JPG.LoadFromFile "C:\images\a.jpg"
```

These instructions show methods without brackets surrounding the parameters.

Assigning a property value requires an equals sign. Missing the equals sign also results in the error "Object doesn't support this property or method". The correct syntax is:

```
JPG.JPGQuality = 80
```

7.1.1. ASP with Javascript

When using Javascript with ASP brackets are needed around function parameters. The backslash character is used as an escape character in Javascript and two should be used together when a backslash is needed:

```
JPG.LoadFromFile("C:\\images\\a.jpg");
```

7.2. Cold Fusion

In Cold Fusion, a COM object is created using the <cfobject> tag:

<cfobject action="create" name="jpg" class="ASPPhotoResizer.Resize">

Each command must be placed inside a `<cfset>` tag and all method parameters must be enclosed by brackets:

```
<cfset JPG.LoadFromFile("c:\images\big.jpg")>
<cfset JPG.ScaleImage(25)>
<cfset JPG.SaveToFile("c:\images\small.jpg")>
```

Alternatively, the commands can be put inside a `<cfscript>` block:

```
<cfscript>
JPG.LoadFromFile("c:\images\big.jpg");
JPG.ScaleImage(25);
JPG.SaveToFile("c:\images\small.jpg");
</cfscript>
```

Cold Fusion does not support variant arrays and so the *VariantIn*, *VariantOut* and *StreamImage* commands cannot be used. Images cannot be streamed directly to the browser so any dynamically produced image must be saved to a temporary file first and then displayed using a `<cfcontent>` tag.

8. Alphabetical List of Commands