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

csIniFile - ASP IniFile Editing Component from Chestysoft

Introduction

This component provides a comprehensive set of functions for reading, creating and editing Windows INI files from within an ASP script.

The ActiveX file csIniFile.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

Windows INI File Format

The INI file text format is the standard way for Windows 3.x applications to store and retrieve application settings from session to session. An INI file stores information in logical groupings, called "sections". The section name is enclosed in square brackets, for example "[Section1]". Within each section, actual data values are stored in named keys. Keys take the form: `<keyname>=<value>`

Viewing any INI file with a text editor will reveal this structure.

Under 32-bit Windows systems, applications typically use the system registry to store and retrieve their settings, instead of INI files. However, there are still many cases where INI files are used effectively. With this component an INI file can easily be used to store the settings of a web application.

Object Creation

The code:

`Set Obj = Server.CreateObject("csIniFile.UseFile")` - creates an instance of the object called “Obj”.

In the trial version, use:

```
Set Obj = Server.CreateObject("csIniFileTrial.UseFile")
```

Reading and Writing INI Files

The FileName Property

Before reading or writing from an INI file, it is necessary to set the FileName property to the name and full path of the INI file to be read from/written to. If there is no file by that name it will not generate an error. The first attempt to write to a non-existent file will cause the file to be created, if possible.

Example:

```
Obj.FileName = "c:\adirectory\example.ini"
```

This property can be also be read.

Deleting Entries From the INI File

These two methods delete keys and sections.

DeleteKey *Section, KeyName* - Deletes the named key and its value within the named section.

EraseSection *Section* - Deletes the entire named section.

Reading INI File Keys

The following read only properties return the value of the named key from the named section. In each case, a third parameter "Default" is included which is the value returned if the key is not defined.

ReadString(*Section, Keyname, Default*) - Returns a string value.

ReadBool(*Section, KeyName, Default*) - Returns a Boolean value, "True" or "False". A Boolean value is stored in the INI file as 1 for true, and 0 for false. Any non-zero integer value will read as true.

ReadDate(*Section, KeyName, Default*) - Returns a date.

ReadDateTime(*Section, KeyName, Default*) - Returns a date and time.

ReadFloat(*Section, KeyName, Default*) - Returns a floating point number.

ReadInteger(*Section, KeyName, Default*) - Returns an integer number.

ReadTime(*Section, KeyName, Default*) - Returns a time.

Verifying if a Section or Key exists

Two properties are available:

SectionExists(*Section*) - Returns true if the section exists.

ValueExists(*Section, KeyName*) - Returns true if the key exists within the named section.

Writing Values to the IniFile

For each of the properties listed for reading a value there is a corresponding method to write a value. Attempting to write data to a non-existent section or a non-existent key is not an error. The section and/or key will be created and the new value set.

WriteString *Section, KeyName, Value* - Writes a string value.

WriteBool *Section, KeyName, Value* - Writes a Boolean value.

WriteDate *Section, KeyName, Value* - Writes a date.

WriteDateTime *Section, KeyName, Value* - Writes a DateTime.

WriteFloat *Section, KeyName, Value* - Writes a floating point number.

WriteInteger *Section, KeyName, Value* - Writes an integer number.

WriteTime *Section, KeyName, Value* - Writes a time.

The Sections and SectionKeys Collections

There are two "collections" provided to read all the sections in an INI file, and all the keys in a section. These can be looped through using the For..Each construct. Both collections have Item and Count properties. Both collections are read only so there is no Add method and Item cannot be written to.

The Sections Collection

Before reading values from the collection, it is necessary to read the values into it. This takes the form:

Sections.ReadSections

Example:

```
Obj.Sections.ReadSections
```

This assumes an object called Obj has been created and the FileName property has been set. It will fill the Sections collection with the names of all the sections in the INI file.

The number of sections is returned by:

```
Obj.Sections.Count
```

The name of the first section is returned by:

```
Obj.Sections.Item(0)
```

Note that it is a zero based array.

The SectionKeys Collection

This must also be initialised before reading from it. This takes the form:

SectionKeys.ReadKeys *Section*

Example:

```
Obj.SectionKeys.ReadKeys "Section1"
```

This assumes an object called Obj has been created and the FileName property has been set. It will fill the SectionKeys collection with the names of all the keys in the section called "Section1".

This example shows how to list all the key names in each section of an INI file.

```
<%
  Set Obj = Server.CreateObject("csIniFile.UseFile")
  Obj.FileName = "c:\adirectory\example.ini"
  Obj.Sections.ReadSections
  For Each Section in Obj.Sections
    Response.Write Section & "<br>"
    Obj.SectionKeys.ReadKeys Section
    For Each Key in Obj.SectionKeys
      Response.Write "&nbsp;&nbsp;" & Key & "<br>"
    Next
  Next
%>
```

The file name must be set first. The Sections collection is set before looping through it. The SectionKeys collection is set for each section, and then the loop finds each key name. This example does not read any values.

File Utilities

There are a number of file utility functions included for convenience. They are not intended to be a comprehensive set, because standard asp has the File Access component to cover most file utilities. These are functions which may be useful while dealing with INI files.

CurrentDir - This property returns the actual 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 directory path. The return value is the parent directory.

Example:

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

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

Scriptname - A read only property returning the current script name complete with extension.

FileSize(*FileName*) - *FileName* is the full path and filename of a file. The return value is the file size in bytes.

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:

```
	Obj.AppendToFile Obj.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 directly. It is useful for maintaining a simple log file. 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 Version Property

Version - This is a read only property showing the version of the csIniFile.dll file. In the case of the trial component, it shows the expiry date.

The Style Sheet Demo

This demonstration application is supplied in the standard download. It consists of four asp files, "control.asp", "style.asp", "samplepage.asp" and "showini.asp". These four files must be in the same directory, which must be web shared with permission to run scripts. When the application is run it will produce an INI file called "style.asp". Control.asp is the starting page.

This application shows how web site settings can be stored in an INI file and retrieved as required. It stores the size and fonts for three html headings and for paragraphs. The control page writes the values to the INI file. The page "style.asp" is used as an include file. It reads the values from the INI file and produces a variable called "Style" which forms the body of a <style> tag for an html page. SamplePage.asp calls the include file and uses the style settings when the page is displayed.

ShowIni.asp uses the Sections and SectionKeys collections and a nested For..Each loop to display the entire file.

The demonstration assumes the the trial version, csIniFileTrial is being used. The object creation parameter would need to be modified to work with the full version, csIniFile.

Other Products From Chestysoft

Visit the Chestysoft web site for details of other components.

ActiveX Controls:

csXImage - ActiveX control to display, edit and scan images.

csXGraph - ActiveX control to draw pie charts, bar charts and line graphs.

csXPostUpload - ActiveX control to upload batches of files to a server using HTTP.

csXThumbUpload - Upload multiple files by HTTP or FTP with previews and image edits.

csXMultiUpload - Select and upload multiple files and post to a server using HTTP.

Active Server Page Components:

csImageFile - Resize, edit and join JPG, BMP, PNG, GIF and other images.

csASPUpload - ASP component to upload files through a browser form.

csASPZipFile - Create ZIP files and control downloads.

csFileDownload - ASP component to control file downloads from a script.

csDrawGraph - Draw pie charts, bar charts and line graphs.

csASPGif - Create and edit animated GIFs.

csFTPQuick - ASP component to transfer files using FTP.

ASP.NET Components

csASPNetGraph - A .NET component to draw pie charts, bar charts and line graphs.

csNetUpload - ASP.NET component for saving HTTP uploads.

csNetDownload - ASP.NET class to control file downloads.

Web Hosting

We can offer ASP enabled web hosting with our components installed. Click for more details.

Chestysoft, July 2010.
www.chestysoft.com