Gets information about the files in a directory and puts it in an array of strings.

```
#region License and Copyright
/* ————————————————————————-
* Dotnet Commons IO
*
*
* This library is free software; you can redistribute it and/or modify it
* under the terms of the GNU Lesser General Public License as published by
* the Free Software Foundation; either version 2.1 of the License, or
* (at your option) any later version.
*
* This library is distributed in the hope that it will be useful, but
* WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
* or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License
* for more details.
*
* You should have received a copy of the GNU Lesser General Public License
* along with this library; if not, write to the
*
* Free Software Foundation, Inc.,
* 59 Temple Place,
* Suite 330,
* Boston,
* MA 02111-1307
```

```
* USA
*
* ---------------------------
*/
#endregion

using System;
using System.Collections;
using System.Globalization;
using System.IO;

namespace Dotnet.Commons.IO
{

///
///

/// This class provides basic facilities for manipulating
files and file paths.
///
///
```

File-related methods

```
/// There are methods to
///  /// copy a file to another file,
/// compare the content of 2 files,
/// delete files using the wildcard character,
/// etc
///  ///

///
public sealed class FileUtils
{
/// ----------------------
///

/// Gets information about the files in a directory and puts
it in an array of strings.
/// The file attributes are separated by commas.
```

```
///
///  ///
/// An string array containing comma separated values of
/// file information in a given directory
///
/// As the comma character is a valid character in a file
name,
/// the values are encapsulated within a double
///  quote,  eg.  “Dotnet.Commons.IO.dll”,”28672″,”26/01/2006
2:25:26   AM”,”27/07/2006   10:18:04   PM”,”27/07/2006
10:16″,”Archive”
///
/// ————————————————————
public static string[] GetDirectoryFileInfo(string directory)
{
return GetDirectoryFileInfo(directory, ',');
}
private static string encapsulateInQuote(string value, bool
toEncapsulate)
{
if (toEncapsulate)
return string.Format(“”{0}””, value);
else
return value;
}
/// ————————————————————
///

/// Get an array of files info from a directory.
///
///  ///
/// ————————————————————
public static FileInfo[] GetFilesInDirectory(string directory)
{
if ((directory == null) || (directory.Length < 1)) throw new
System.ArgumentException("Directory supplied is either null or
empty"); DirectoryInfo dirInfo = new DirectoryInfo(directory);
```

```
if (!dirInfo.Exists) throw new System.ArgumentException("Directory '" + directory + "' does not exist."); return dirInfo.GetFiles(); } /// ---------- ---------- ///
/// Gets information about the files in a directory and puts it in an array of strings.
///

/// name of directory to search /// delimeter character to separator file attributes ///
/// An string array containing comma separated values of
/// file information in a given directory in the format:
///


                                                                ///
filename,Size,CreationTime,LastAccessTime,LastWriteTime,Attributes
        ///

/// assuming that the delimeter chosen is the comma ',' character.
///
/// ———————————————
public static string[] GetDirectoryFileInfo(string directory, char delimeter)
{
ArrayList al = new ArrayList();
al.Add(String.Format(“Name{0}Size{0}CreationTime{0}LastAccessTime{0}LastWriteTime{0}Attributes”, delimeter));

bool toEncapsulateInQuote = delimeter == ',';

FileInfo[] files = GetFilesInDirectory(directory);
for (int i = 0; i < files.Length; i++) { System.Text.StringBuilder buffy = new System.Text.StringBuilder();
buffy.Append(encapsulateInQuote(files[i].Name, toEncapsulateInQuote)); buffy.Append(delimeter);
```

```
buffy.Append(encapsulateInQuote(files[i].Length.ToString(), toEncapsulateInQuote));      buffy.Append(delimeter); buffy.Append(encapsulateInQuote(files[i].CreationTime.ToString(),   toEncapsulateInQuote));   buffy.Append(delimeter); buffy.Append(encapsulateInQuote(files[i].LastAccessTime.ToString(),   toEncapsulateInQuote));   buffy.Append(delimeter); buffy.Append(encapsulateInQuote(files[i].LastWriteTime.ToString(),   toEncapsulateInQuote));   buffy.Append(delimeter); buffy.Append(encapsulateInQuote(files[i].Attributes.ToString(), toEncapsulateInQuote)); al.Add(buffy.ToString()); } string[] dInfo = new string[al.Count]; al.CopyTo(dInfo); return dInfo; } } } [/csharp]
```