Working with Directories in C#

Mahesh Chand

July 2012, Garnet Valley PA

Message from the Author

Thank you for being a part of C# Corner, a free online community for IT developers and professionals.

I've always been a big believer and advocate of free knowledge sharing and education to all. C# Corner is all about helping each other.

I will have to say a big thank you to you and our top members who made all this possible. This is all possible because of you and you believing in sharing your code and knowledge.

Please feel free to share this book with your friends and co-workers and do not forget to share your knowledge and spread the word around about C# Corner and the Mindcracker Network.

Namaste!

Mahesh Chand

Microsoft MVP, Visual C#
Founder, C# Corner and Mindcracker Network

Introduction

Input and output (I/O) streaming is a process of reading data from and writing data to a storage medium. In the .NET Framework, the System.IO namespace and its sub namespaces contain the classes and other types to perform I/O operations.

The Directory class in the .NET Framework class library provides static methods for creating, reading, copying, moving, and deleting directories.

This book covers the following topics:

- Understand the Directory class
- Directory properties
- How to create a directory
- How create a subdirectory
- How to read all directory and directories
- How to read all files in a directory
- How to move a directory
- How to delete a directory

Directory

The System.IO.Directory class in the .NET Framework class library provides static methods for creating, copying, moving, and deleting directories and subdirectories. Before you can use the Directory class, you must import the System.IO namespace.

```
using System.IO;
```

Create a Directory

The Directory.CreateDirectory method creates a directory in the specified path with the specified Windows security. You can also create a directory on a remote compute.

The following code snippet creates a Temp folder in C:\ drive if the directory does not exists already.

```
string root = @"C:\Temp";
string subdir = @"C:\Temp\Mahesh";
// If directory does not exist, create it.
if (!Directory.Exists(root))
{
    Directory.CreateDirectory(root);
}
```

The Directory.CreateDirectory is also used to create a sub directory. All you have to do is to specify the path of the directory in which this subdirectory will be created in. The following code snippet creates a Mahesh subdirectory in C:\Temp directory.

```
// Create sub directory
if (!Directory.Exists(subdir))
{
    Directory.CreateDirectory(subdir);
}
```

Delete a directory in C#

The Directory.Delete method deletes an empty directory from the specified path permanently. If a directory has subdirectories and/or files, you must delete them before you can delete a directory. If you try to delete a file that is not empty, you will get an error message.

The following code snippet deletes the destination file.

```
string root = @"C:\Temp";
// If directory does not exist, don't even try
if (Directory.Exists(root))
{
    Directory.Delete(root);
}
```

The following code snippet checks if a directory has subdirectories and files and delete them before deleting a directory.

Check if a directory Exists

The Directory.Exists method checks if the specified directory exists. The following code snippet checks if a directory exists or not and deletes only if the directory exists.

```
string root = @"C:\Temp";
// If directory does not exist, don't even try
if (Directory.Exists(root))
{
    Directory.Delete(root);
}
```

Move a directory in C#

The Directory.Move method moves an existing directory to a new specified directory with full path. The Move method takes two parameters. The Move method deletes the original directory.

The following code snippet moves the source directory to the destination directory.

```
string sourceDirName = @"C:\Temp";
string destDirName = @"C:\NewTemp";
try
{
    Directory.Move(sourceDirName, destDirName);
}
catch (IOException exp)
{
    Console.WriteLine(exp.Message);
}
```

Copy a directory in C#

There is no method to copy a directory. The copying a directory is a process of creating a new directory that you to want a directory to move to and then copying subdirectory and files.

Get and Set Directory Creation Time

The SetCreationTime and GetCreationTime methods are used to set and get the creation date and time of the specified file. The following code snippet sets and gets the creation time of a file.

```
// Get and set file creation time
string fileName = @"c:\temp\Mahesh.txt";
File.SetCreationTime(fileName, DateTime.Now);
DateTime dt = File.GetCreationTime(fileName);
Console.WriteLine("File created time: {0}", dt.ToString());
```

Get and Set File Last Access Time

The SetLastAccessTime and GetLastAccessTime methods are used to set and get the last access date and time of the specified file. The following code snippet sets and gets the last access date and time of a file.

```
// Get and set file last access time
string fileName = @"c:\temp\Mahesh.txt";
File.SetLastAccessTime(fileName, DateTime.Now);
DateTime dt = File.GetLastAccessTime(fileName);
Console.WriteLine("File last access time: {0}", dt.ToString());
```

Get and Set File Last Write Time

The SetLastWriteTime and GetLastWriteTime methods are used to set and get the last write date and time of the specified file. The following code snippet sets and gets the last write date and time of a file.

```
// Get and set file last write time
string fileName = @"c:\temp\Mahesh.txt";
File.SetLastWriteTime(fileName, DateTime.Now);
DateTime dt = File.GetLastWriteTime(fileName);
Console.WriteLine("File last write time: {0}", dt.ToString());
```

Enumerate Directory in C#

The Directory.EnumerateDirectories method returns an enumerable collection of directory names in the specified directory.

```
string root = @"C:\Temp";

// Get a list of all subdirectories
var dirs = from dir in
                Directory.EnumerateDirectories(root)
            select dir;
Console.WriteLine("Subdirectories: {0}", dirs.Count<string>().ToString());
Console.WriteLine("List of Subdirectories");
foreach (var dir in dirs)
{
    Console.WriteLine("{0}",
        dir.Substring(dir.LastIndexOf("\\") + 1));
}

// Get a list of all subdirectories starting with 'Ma'
var MaDirs = from dir in
                Directory.EnumerateDirectories(root, "Ma*")
            select dir;
Console.WriteLine("Subdirectories: {0}", MaDirs.Count<string>().ToString());
Console.WriteLine("List of Subdirectories");
foreach (var dir in MaDirs)
{
    Console.WriteLine("{0}",
            dir.Substring(dir.LastIndexOf("\\") + 1));
}
```

Enumerate Files in C#

The Directory.EnumerateFiles method returns an enumerable collection of file names in the specified directory.

```
string root = @"C:\Temp";

// Get a list of all subdirectories
var files = from file in
                Directory.EnumerateFiles(root)
            select file;
Console.WriteLine("Files: {0}", files.Count<string>().ToString());
Console.WriteLine("List of Files");
foreach (var file in files)
{
    Console.WriteLine("{0}", file);
}
```

Get and Set File Creation Time

The Directory.SetCreationTime and Directory.GetCreationTime methods are used to set and get the creation date and time of the specified directory. The following code snippet sets and gets the creation time of a directory.

```
string root = @"C:\Temp";
// Get and Set Creation time
Directory.SetCreationTime(root, DateTime.Now);
DateTime creationTime = Directory.GetCreationTime(root);
Console.WriteLine(creationTime);
```

Get and Set File Last Access Time

The SetLastAccessTime and GetLastAccessTime methods are used to set and get the last access date and time of the specified directory. The following code snippet sets and gets the last access date and time of a directory.

```
string root = @"C:\Temp";
// Get and Set Last Access time
Directory.SetLastAccessTime(root, DateTime.Now);
DateTime lastAccessTime = Directory.GetLastAccessTime(root);
Console.WriteLine(lastAccessTime);
```

Get and Set File Last Write Time

The SetLastWriteTime and GetLastWriteTime methods are used to set and get the last write date and time of the specified directory. The following code snippet sets and gets the last write date and time of a directory.

```
string root = @"C:\Temp";
// Get and Set Last Write time
Directory.SetLastWriteTime(root, DateTime.Now);
DateTime lastWriteTime = Directory.GetLastWriteTime(root);
Console.WriteLine(lastWriteTime);
```

Get and Set Current Directory in C#

The SetCurrentDirectory method sets the specified directory as the current directory. The GetCurrentDirectory method returns the current directory.

```
string root = @"C:\Temp";
Directory.SetCurrentDirectory(root);
```

```
Console.WriteLine(Directory.GetCurrentDirectory());
```

Get Sub Directories in C#

The GetDirectories method of the Directory class loads all the subdirectories of a directory. To get all subdirectories, we can read subdirectories recursively.

```
public void GetSubDirectories()
{
    string root = @"C:\Temp";
    // Get all subdirectories
    string[] subdirectoryEntries = Directory.GetDirectories(root);
    // Loop through them to see if they have any other subdirectories
    foreach (string subdirectory in subdirectoryEntries)
        LoadSubDirs(subdirectory);
}
private void LoadSubDirs(string dir)
{
    Console.WriteLine(dir);
    string[] subdirectoryEntries = Directory.GetDirectories(dir);
    foreach (string subdirectory in subdirectoryEntries)
    {
        LoadSubDirs(subdirectory);
    }
}
```

Get Files in a Directory in C#

The GetFiles method gets a list of files in the specified directory.

```
string root = @"C:\Temp";
string[] fileEntries = Directory.GetFiles(root);
foreach (string fileName in fileEntries)
    Console.WriteLine(fileName);
```

Get Root Directory in C#

The GetRootDirecoty method returns the root directory of the specified directory.

```
string root = @"C:\Temp";
Console.WriteLine(Directory.GetDirectoryRoot(root));
```

Get all drives in C#

The GetLogicalDrives method returns all the logical drives on a system.

```
string[] drives = System.IO.Directory.GetLogicalDrives();
foreach (string drive in drives)
{
    System.Console.WriteLine(drive);
}
```

Help Us

There are many ways you can help us grow this free community.

- Spread the word around by letting your co-workers and friends know about C# Corner.
- Share an article or book links with them.
- By posting your articles, code snippets, tips and blogs online >>
- By answering questions on C# Corner Answers >>
- By sharing your Interview questions or posting interview answers >>
- By sharing software and IT related news here >>
- By simply visiting C# Corner and letting us know how we are doing and what more we have to do to improve our services.

Download more free books on C# Corner books section here >>