FileInfo 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. There is a saying in Hindi, AKELA CHANA BHAD NAHIN FOD SAKTA. 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 FileInfo class in the .NET Framework class library provides static methods for creating, reading, copying, moving, and deleting files using the FileStream objects.

This book covers the following topics:

- Understand the FileInfo class
- FileInfo properties
- How to create a file
- How to open and read files
- How to write to the files
- How to move, replace and delete files
- How to encrypt and decrypt files

Create a FileInfo

A FileInfo object is created using the default constructor that takes a string as a file name with a full path.

```
string fileName = @"C:\Temp\MaheshTXFI.txt";
FileInfo fi = new FileInfo(fileName);
```

FileInfo Properties

The FileInfo class provides properties to get file name, extension, directory, size and file attributes.

Get File Name

The FileName property returns just the file name part of the full path of a file. The following code snippet returns the file name.

```
string justFileName = fi.Name;
Console.WriteLine("File Name: {0}", justFileName);
```

Get Full Path of a File

The FullName property returns just the full path of a file including the file name. The following code snippet returns the full path of a file.

```
string fullFileName = fi.FullName;
Console.WriteLine("File Name: {0}", fullFileName);
```

Get a File Extension

The Extension property returns the extension of a file. The following code snippet returns the extension of a file.

```
string extn = fi.Extension;
Console.WriteLine("File Extension: {0}", extn);
```

Get Directory Name of a File

The DirectoryName property returns the name of the directory of a file. The following code snippet returns the directory of a file.

```
string directoryName = fi.DirectoryName;
```

```
Console.WriteLine("Directory Name: {0}", directoryName);
```

Check if a File Exists

The Exists property returns true if a file exists. The following code snippet returns true if a file already exists.

```
bool exists = fi.Exists;
```

Get a file size

The Length property returns the size of a file in bytes. The following code snippet returns the size of a file.

```
// Get file size
long size = fi.Length;
Console.WriteLine("File Size in Bytes: {0}", size);
```

Check if a file is read only

The IsReadOnly property returns if a file is read only. The following code snippet returns true if a file is read only.

```
// File ReadOnly ?
bool IsReadOnly = fi.IsReadOnly;
Console.WriteLine("Is ReadOnly: {0}", IsReadOnly);
```

File Creation Time

The CreationTime property returns the DateTime when a file was creaed. The following code snippet returns the creation time of a file.

```
// Creation, last access, and last write time
DateTime creationTime = fi.CreationTime;
Console.WriteLine("Creation time: {0}", creationTime);
```

File Last Access Time

The LastAccessTime property returns the DateTime when a file was last accessed. The following code snippet returns the last access time of a file.

```
DateTime accessTime = fi.LastAccessTime;
Console.WriteLine("Last access time: {0}", accessTime);
```

File Last Updated Time

The LastWriteTime property returns the DateTime when a file was last updated or written. The following code snippet returns the last write time of a file.

```
DateTime updatedTime = fi.LastWriteTime;
Console.WriteLine("Last write time: {0}", updatedTime);
```

Sample

Here is a complete sample.

```
// Full file name
string fileName = @"C:\Temp\MaheshTXFI.txt";
FileInfo fi = new FileInfo(fileName);

// Create a new file
using (FileStream fs = fi.Create())
{
    Byte[] txt = new UTF8Encoding(true).GetBytes("New file.");
    fs.Write(txt, 0, txt.Length);
    Byte[] author = new UTF8Encoding(true).GetBytes("Author Mahesh Chand");
    fs.Write(author, 0, author.Length);
}

// Get File Name
string justFileName = fi.Name;
Console.WriteLine("File Name: {0}", justFileName);
// Get file name with full path
string fullFileName = fi.FullName;
Console.WriteLine("File Name: {0}", fullFileName);
// Get file extension
string extn = fi.Extension;
Console.WriteLine("File Extension: {0}", extn);
// Get directory name
string directoryName = fi.DirectoryName;
Console.WriteLine("Directory Name: {0}", directoryName);
// File Exists ?
bool exists = fi.Exists;
Console.WriteLine("File Exists: {0}", exists);
if (fi.Exists)
{
    // Get file size
    long size = fi.Length;
    Console.WriteLine("File Size in Bytes: {0}", size);
    // File ReadOnly ?
    bool IsReadOnly = fi.IsReadOnly;
    Console.WriteLine("Is ReadOnly: {0}", IsReadOnly);
    // Creation, last access, and last write time
    DateTime creationTime = fi.CreationTime;
    Console.WriteLine("Creation time: {0}", creationTime);
    DateTime accessTime = fi.LastAccessTime;
    Console.WriteLine("Last access time: {0}", accessTime);
    DateTime updatedTime = fi.LastWriteTime;
    Console.WriteLine("Last write time: {0}", updatedTime);
}
```

Create a File

We can create a file in two different following methods

- File.Create
- File.CreateText

FileInfo.Create Method

The FileInfo.Create method creates a file at the given path. If just a file name is provided without a path, the file will be created in the current folder.

The following code snippet creates a file using the Create method that returns a FileSteam object. The Write method of FileStream can be used to write text to the file.

```
// Full file name
string fileName = @"C:\Temp\MaheshTXFI.txt";
FileInfo fi = new FileInfo(fileName);

try
{
    // Check if file already exists. If yes, delete it.
    if (fi.Exists)
    {
        fi.Delete();
    }

    // Create a new file
    using (FileStream fs = fi.Create())
    {
        Byte[] txt = new UTF8Encoding(true).GetBytes("New file.");
        fs.Write(txt, 0, txt.Length);
        Byte[] author = new UTF8Encoding(true).GetBytes("Author Mahesh Chand");
        fs.Write(author, 0, author.Length);
    }

    // Write file contents on console.
    using (StreamReader sr = File.OpenText(fileName))
    {
        string s = "";
        while ((s = sr.ReadLine()) != null)
        {
            Console.WriteLine(s);
        }
    }
}
catch (Exception Ex)
{
    Console.WriteLine(Ex.ToString());
}
```

FileInfo.CreateText Method

The FileInfo.CreateText method creates and opens a file for writing UTF-8 encoded text. If file already exists, this method opens the file.

The following code snippet creates a file using the CreateText method that returns a StreamWriter object. The WriteLine method of SteamLine can be used to add line text to the object and writes to the file.

```
// Full file name
string fileName = @"C:\Temp\MaheshTXFITx.txt";
FileInfo fi = new FileInfo(fileName);

try
{
    // Check if file already exists. If yes, delete it.
    if (fi.Exists)
    {
        fi.Delete();
    }

    // Create a new file
    using (StreamWriter sw = fi.CreateText())
    {
        sw.WriteLine("New file created: {0}", DateTime.Now.ToString());
        sw.WriteLine("Author: Mahesh Chand");
        sw.WriteLine("Add one more line ");
        sw.WriteLine("Add one more line ");
        sw.WriteLine("Done! ");
    }

    // Write file contents on console.
    using (StreamReader sr = File.OpenText(fileName))
    {
        string s = "";
        while ((s = sr.ReadLine()) != null)
        {
            Console.WriteLine(s);
        }
    }
}
catch (Exception Ex)
{
    Console.WriteLine(Ex.ToString());
}
```

Read and Write a File

Open a File

A File must be opened using an IO resource before it can be read or write to. A file can be opened to read and/or write purpose. The FileInfo class provides four methods to open a file.

- Open
- OpenRead
- OpenText
- OpenWrite

File.Open Method

The Open method opens a FileStream on the specified file in the specified file mode.

```
FileStream fs = fi.Open(FileMode.Open, FileAccess.Write);
```

Here is the complete code sample.

```
string fileName = @"C:\Temp\MaheshTXFITx.txt";
FileInfo fi = new FileInfo(fileName);

// If file does not exist, create file
if (!fi.Exists)
{
    //Create the file.
    using (FileStream fs = fi.Create())
    {
        Byte[] info = new UTF8Encoding(true).GetBytes("File Start");
        fs.Write(info, 0, info.Length);
    }
}

try
{
    using (FileStream fs = fi.Open(FileMode.Open, FileAccess.Write))
    {
        Byte[] info = new UTF8Encoding(true).GetBytes("Add more text");
        fs.Write(info, 0, info.Length);
    }
}
catch (Exception Ex)
{
    Console.WriteLine(Ex.ToString());
}
```

File.OpenRead Method

The OpenRead method opens a file for reading. The method returns a FileStream object that is used to read a file using its Read method.

```
string fileName = @"C:\Temp\MaheshTXFITx.txt";
FileInfo fi = new FileInfo(fileName);
FileStream fs = fi.OpenRead();
```

The file is read into a byte array. The following code snippet uses the FileStream.Read method and gets text into a byte array and then it is converted to a string using UTF8Encoding.GetString method.

```
using (FileStream fs = fi.OpenRead())
{
    byte[] byteArray = new byte[1024];
    UTF8Encoding fileContent = new UTF8Encoding(true);
    while (fs.Read(byteArray, 0, byteArray.Length) > 0)
    {
        Console.WriteLine(fileContent.GetString(byteArray));
    }
}
```

The following lists the complete code sample.

```
string fileName = @"C:\Temp\MaheshTXFITx.txt";
FileInfo fi = new FileInfo(fileName);

// If file does not exist, create file
if (!fi.Exists)
{
    //Create the file.
    using (FileStream fs = fi.Create())
    {
        Byte[] info = new UTF8Encoding(true).GetBytes("File Start");
        fs.Write(info, 0, info.Length);
    }
}
try
{
using (FileStream fs = fi.OpenRead())
{
    byte[] byteArray = new byte[1024];
    UTF8Encoding fileContent = new UTF8Encoding(true);
    while (fs.Read(byteArray, 0, byteArray.Length) > 0)
    {
        Console.WriteLine(fileContent.GetString(byteArray));
    }
}
}
catch (Exception Ex)
{
    Console.WriteLine(Ex.ToString());
}
```

File.OpenText Method

The OpenText method opens an existing UTF-8 encoded text file for reading. The OpenText method takes a file name as a parameter and returns a StreamReader object.

```
StreamReader reader = fi.OpenText();
```

Once we have a StreamReader object, we can use its Read or ReadLine methods to read the contents. The ReadLine method reads one line at a time. The following code snippet loops through the entire content of a SteamReader and reads and prints one line at a time.

```
while ((s = reader.ReadLine()) != null)
{
    Console.WriteLine(s);
}
```

The following lists the complete code sample.

```
string fileName = @"C:\Temp\MaheshTXFITx.txt";
FileInfo fi = new FileInfo(fileName);

// If file does not exist, create file
if (!fi.Exists)
{
    //Create the file.
    using (FileStream fs = fi.Create())
    {
        Byte[] info = new UTF8Encoding(true).GetBytes("File Start");
        fs.Write(info, 0, info.Length);
    }
}

try
{
    using (StreamReader reader = fi.OpenText())
    {
        string s = "";
        while ((s = reader.ReadLine()) != null)
        {
            Console.WriteLine(s);
        }
    }
}
catch (Exception Ex)
{
    Console.WriteLine(Ex.ToString());
}
```

File.OpenWrite Method

The OpenWrite method opens a file for writing. If file does not exist, it creates a new file and opens if for writing. The OpenWrite method takes a file name as parameter and returns a FileStream object on the specified file.

```
FileStream fs = fi.OpenWrite();
```

Once we have a FileStream object, we can use its Write method to write to the file. The WriteMethod takes a byte array. The following code snippet creates a byte array and passes it to the Write method of the FileStream.

```
Byte[] info = new UTF8Encoding(true).GetBytes("New File using OpenWrite Method \n");
fs.Write(info, 0, info.Length);
```

The following lists the complete code sample.

```
string fileName = @"C:\Temp\MaheshTXFITx.txt";
FileInfo fi = new FileInfo(fileName);

// If file does not exist, create file
if (!fi.Exists)
{
    //Create the file.
    using (FileStream fs = fi.Create())
    {
        Byte[] info = new UTF8Encoding(true).GetBytes("File Start");
        fs.Write(info, 0, info.Length);
    }
}

try
{
    // Open a file and add some contents to it
    using (FileStream fs = fi.OpenWrite())
    {
        Byte[] info = new UTF8Encoding(true).GetBytes("New File using OpenWrite Method
\n");
        fs.Write(info, 0, info.Length);
        info = new UTF8Encoding(true).GetBytes("----------START------------------------
\n");
        fs.Write(info, 0, info.Length);
        info = new UTF8Encoding(true).GetBytes("Author: Mahesh Chand \n");
        fs.Write(info, 0, info.Length);
        info = new UTF8Encoding(true).GetBytes("Book: ADO.NET Programming using C#\n");
        fs.Write(info, 0, info.Length);
        info = new UTF8Encoding(true).GetBytes("----------END------------------------");
        fs.Write(info, 0, info.Length);
    }
    // Read file contents and display on the console
    using (FileStream fs = File.OpenRead(fileName))
    {
        byte[] byteArray = new byte[1024];
        UTF8Encoding fileContent = new UTF8Encoding(true);
        while (fs.Read(byteArray, 0, byteArray.Length) > 0)
```

```
            {
                Console.WriteLine(fileContent.GetString(byteArray));
            }
        }
}
catch (Exception Ex)
{
    Console.WriteLine(Ex.ToString());
}
```

Append a File

The AppendText method creates a StreamWriter object that appends UTF-8 encoded text to an existing text file.

```
string fileName = @"C:\Temp\Data\MaheshTXMoved.txt";
FileInfo fi = new FileInfo(fileName);

using (StreamWriter sw = fi.AppendText())
{
    sw.WriteLine("--------- Append Text Start ----------");
    sw.WriteLine("New author started");
    sw.WriteLine("a book on Files Programming ");
    sw.WriteLine("using C#");
    sw.WriteLine("--------- Append Text End ----------");
}
// Read all text
string readText = File.ReadAllText(fileName);
Console.WriteLine(readText);
```

Copy a File in C#

The Copy method copies an existing file to a new file on the specified location. The Copy method takes two parameters. First the file name to be copied to with full and the second parameter that is optional that is used to overwrite an existing file. If the second parameter is true, the Copy method will overwrite if file already exists.

The following code snippet copies a file to the destination file.

```
string fileName = @"C:\Temp\MaheshTXFITx.txt";
FileInfo fi = new FileInfo(fileName);
string destinationFile = @"C:\Temp\Data\MaheshTXCopied.txt";
try
{
    fi.CopyTo(destinationFile, true);
}
```

```
catch (IOException iox)
{
    Console.WriteLine(iox.Message);
}
```

Move a File in C#

The Move method moves an existing file to a new location with the same or a different file name. The Move method takes the full path of the move file. The Move method deletes the original file.

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

```
string fileName = @"C:\Temp\MaheshTXFITx.txt";
FileInfo fi = new FileInfo(fileName);
string destinationFile = @"C:\Temp\Data\MaheshTXMoved.txt";
try
{
    fi.MoveTo(destinationFile);
}
catch (IOException iox)
{
    Console.WriteLine(iox.Message);
}
```

Replace a File in C#

The Replace method replaces the contents of a specified file with the contents of another file. This method deletes the original file and creates a backup of the replaced file.

The following code snippet moves the contents of the original file into the replaced file and also creates a backup of the replaced file and deletes the original file.

```
string repFile = @"C:\Temp\MaheshTXFI.txt";
string backupFile = @"C:\Temp\MaheshTXFI.txt.bac";
string fileName = @"C:\Temp\MaheshTXFITx.txt";
FileInfo fi = new FileInfo(fileName);

try
{
    fi.Replace(repFile, backupFile, false);
}
catch (IOException iox)
{
    Console.WriteLine(iox.Message);
}
```

Delete a File in C#

The Delete method deletes the specified file permanently. The following code snippet deletes a file.

```
string fileName = @"C:\Temp\Data\MaheshTXMoved.txt";
FileInfo fi = new FileInfo(fileName);
try
{
    fi.Delete();
}
catch (IOException iox)
{
    Console.WriteLine(iox.Message);
}
```

Encrypt a File in C#

The Encrypt method encrypts a file so that only the account used to encrypt the file can decrypt it.

```
string fileName = @"C:\Temp\Data\MaheshTXMoved.txt";
FileInfo fi = new FileInfo(fileName);
fi.Encrypt();
fi.Decrypt();
```

Decrypt a File in C#

The Decrypt method decrypts an encrypted file. Only account that has encrypted a file can decrypt a file.

```
string fileName = @"C:\Temp\Data\MaheshTXMoved.txt";
FileInfo fi = new FileInfo(fileName);
fi.Encrypt();
fi.Decrypt();
```

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 >>