File Handling in C#

By

Narasimha Rao T

Microsoft.Net FSD Trainer

Professional Development Trainer

tnrao.trainer@gmail.com

Introduction to File Handling

File handling in C# allows you to:

- Create, read, write, delete, and manipulate files and directories.
- Work with text, binary, or structured data.
- Access files using **high-level** classes from `System.IO`.

Namespaces Required

```
using System.IO;
```

This namespace contains all file handling-related classes like `File`, `FileInfo`, `StreamReader`, `StreamWriter`, `Directory`, etc.

File vs FileInfo Classes

Feature	`File` Class (Static)	`FileInfo` Class (Instance-Based)
Type	Static Class	Non-Static (Object-oriented)
Performance	Slower (new security checks)	Faster on multiple operations
Usage Style	Direct static method calls	Create an object and then operate

The `File` Class

The `File` class provides static methods for:

- Creating, copying, deleting, moving, opening files.

Common Methods:

```
File.Create(path);
File.Copy(sourcePath, destPath);
File.Delete(path);
File.Exists(path);
File.ReadAllText(path);
File.WriteAllText(path, "Hello World");
```

Reading from a Text File

Using `File.ReadAllText()`

```
string content = File.ReadAllText("log.txt");
Console.WriteLine(content);
```

Using `StreamReader`

```
using (StreamReader reader = new StreamReader("log.txt"))
{
    string line;
    while ((line = reader.ReadLine()) != null)
    {
        Console.WriteLine(line);
    }
}
```

Writing to a Text File

Using `File.WriteAllText()`

```csharp
File.WriteAllText("log.txt", "Log entry at " + DateTime.Now);
```

Using `StreamWriter`

```csharp
using (StreamWriter writer = new StreamWriter("log.txt", append: true))
{
    writer.WriteLine("Another log entry at " + DateTime.Now);
}
```

Appending Data

- Use `File.AppendAllText()` or `StreamWriter` with append = true.

```
File.AppendAllText("log.txt", "Appended line\n");
```

FileInfo Class

```csharp
FileInfo file = new FileInfo("data.txt");

// Create file
using (StreamWriter sw = file.CreateText())
{
    sw.WriteLine("Hello FileInfo!");
}


---
// Check properties
Console.WriteLine(file.FullName);
Console.WriteLine(file.Length);
Console.WriteLine(file.Extension);
```

Directory and DirectoryInfo Classes

`Directory` Class (Static)

```
Directory.CreateDirectory("Logs");
string[] files = Directory.GetFiles("Logs");
```

DirectoryInfo Class (Object-Oriented)

```
DirectoryInfo dir = new DirectoryInfo("Logs");

// Create
if (!dir.Exists)
    dir.Create();

// List files
FileInfo[] files = dir.GetFiles();
foreach (FileInfo file in files)
{
    Console.WriteLine(file.Name);
}
```

Working with Stream Classes

Introduction

- **StreamWriter and StreamReader** are part of the **System.IO** namespace and are used for writing to and reading from text files using streams.
- **StreamWriter** – Writes text to a file.
- **StreamReader** – Reads text from a file.
- **Note**: These classes are more efficient for large text operations than File.WriteAllText() or File.ReadAllText().

StreamWriter – Writing to Text Files

StreamWriter writer = new StreamWriter(filePath, append);

- filePath: Path to the file.
- append: true to append, false to overwrite (default is false).

```
using (StreamWriter writer = new StreamWriter("log.txt"))
{
    writer.WriteLine("First log entry.");
    writer.WriteLine("Second log entry.");
}
```

StreamWriter – Writing to Text Files

```
StreamReader reader = new StreamReader(filePath);


using (StreamReader reader = new StreamReader("log.txt"))
{
    string content = reader.ReadToEnd();
    Console.WriteLine(content);
}
```

Read line by line

```
using (StreamReader reader = new StreamReader("log.txt"))
{
    string line;
    while ((line = reader.ReadLine()) != null)
    {
        Console.WriteLine(line);
    }
}
```

Practical Applications

- Logging application events to `.log` files
- Saving user settings in `.txt` files
- Processing input/output for data files
- Reading configuration from flat files

Best Practices

- Always close or dispose streams (`using` blocks recommended).
- Check for file existence before reading.
- Use `Path.Combine()` for cross-platform file paths.
- Handle exceptions using try-catch blocks (`IOException`, `UnauthorizedAccessException`).

Summary

Topic	Key Point
`File`	Static class for quick file operations
`FileInfo`	OOP-based, reusable for file metadata
`StreamReader/Writer`	Efficient for reading/writing text
`FileStream`	Low-level binary data access
`Directory/DirectoryInfo`	Work with directories and folders

Q & A

Narasimha Rao T
tnrao.trainer@gmail.com