Files in C#

In C#, working with files is an essential part of many applications, particularly when it comes to reading from or writing to files. The System.IO namespace provides various classes to work with files, such as File, StreamReader, StreamWriter, FileStream, and others. Here's a detailed explanation of how to handle files in C#, along with examples and code snippets.

1. Reading from a File

Using StreamReader

The StreamReader class is used to read characters from a byte stream in a particular encoding. It's commonly used to read text files.

Example:

```
using System;
using System.IO;

class Program
{   static void Main()
  {
    string path = "example.txt";
    // Ensure the file exists
    if (File.Exists(path))
    {
      // Reading the file using StreamReader
      using (StreamReader reader = new StreamReader(path))
      {
        string content = reader.ReadToEnd(); // Reads the entire file
        Console.WriteLine(content);
      }
    }
    else
    {  Console.WriteLine("File not found.")  }
  }}
```

- **Explanation:** This code checks if a file named example.txt exists. If it does, it reads the entire content of the file and prints it to the console.

2. Writing to a File

Using StreamWriter

StreamWriter is used to write characters to a stream, often used for writing text to a file.

Example:

```
using System;
using System.IO;
class Program
{ static void Main()
  {
    string path = "example.txt";
    // Writing to a file using StreamWriter
    using (StreamWriter writer = new StreamWriter(path))
    {
      writer.WriteLine("Hello, World!");
      writer.WriteLine("Welcome to file handling in C#.");
    }
    Console.WriteLine("Text written to file.");
  }
}
```

- **Explanation:** This code writes two lines of text to example.txt. If the file doesn't exist, StreamWriter will create it. If it does exist, it will be overwritten.

---

3. Appending to a File

Using StreamWriter with append parameter

You can append text to an existing file using StreamWriter by specifying true for the append parameter.

Example:

```
using System;
using System.IO;
class Program
{  static void Main()
  {
    string path = "example.txt";
```

```
    // Appending to a file using StreamWriter
    using (StreamWriter writer = new StreamWriter(path, true))
    {
        writer.WriteLine("This line is appended to the file.");
    }
    Console.WriteLine("Text appended to file.");
  }
}
```

- **Explanation:** This code appends a line of text to the end of example.txt without overwriting the existing content.

---

4. Working with Binary Files

Using FileStream

For binary files, FileStream is often used. It provides access to files for reading and writing bytes.

Example:

```
    static void WriteList(string path, List<int> InputList)
    {
        // Serialize the list of tuples to JSON
        string jsonString = JsonSerializer.Serialize(InputList);
        // Write the JSON string to a file
        File.WriteAllText(path, jsonString);

        Console.WriteLine("Binary data written to file.");
    }
    static List<int> ReadList(string path)
    {
        // Read the text from the file
        string output = File.ReadAllText(path);
        // Deserialize the JSON string back to a list of integers
        return JsonSerializer.Deserialize<List<int>>(output);
    }
```

- **Explanation:** This code writes an array of bytes to a binary file , then reads it back and prints the byte values.

5. File Information and Management

Getting File Information

You can retrieve information about a file using the FileInfo class.

Example:

```
using System;
using System.IO;
class Program
{   static void Main()
  {
    string path = "example.txt";
    FileInfo fileInfo = new FileInfo(path);
    if (fileInfo.Exists)
    {
      Console.WriteLine($"File Name: {fileInfo.Name}");
      Console.WriteLine($"File Size: {fileInfo.Length} bytes");
      Console.WriteLine($"File Extension: {fileInfo.Extension}");
      Console.WriteLine($"File Creation Time: {fileInfo.CreationTime}");
    }
    else
    {
      Console.WriteLine("File not found.");
    }  }}
```

- **Explanation:** This example retrieves and displays various attributes of example.txt, such as its size, extension, and creation time.

6. File Management

Creating and Deleting Files

You can create and delete files using methods in the File class.

Example:

```
using System;
using System.IO;
class Program
{  static void Main()
  {
```

```
        string path = "example.txt";
        // Create a new file
        if (!File.Exists(path))
        {
            File.Create(path).Close(); // Create the file and close it immediately
            Console.WriteLine("File created.");
        }
        // Delete the file
        if (File.Exists(path))
        {
            File.Delete(path);
            Console.WriteLine("File deleted.");
        }
    }
}
```

- **Explanation:** This example creates a file named example.txt and then deletes it if it exists.

Summary

- **StreamReader and StreamWriter** are used for reading and writing text files.
- **FileStream** is used for handling binary files.
- **FileInfo** and **DirectoryInfo** classes provide detailed information and management functions for files and directories.