File IO and Reflection

File IO :

File handling is a very crucial and important feature for many enterprise applications around us. To support this feature Microsoft .NET Framework offers the System.IO namespace, that provides various classes to enable the developers to do I/O.

Objectives:

Using the File class for reading and writing data.
Using the File and FileInfo class to manipulate files.
Using the DirectoryInfo and Directory classes to manipulate directories.

The File class of the System.IO namespace offers various static methods that enable a developer to do direct reading and writing of files. Typically, to read data from a file, you:

1. Get a hold on the file handle.
2. Open a stream to the file.
3. Buffer the file data into memory.
4. Release the hold of file handle once done.

Reading Data from Files : data is read from a file

The "ReadAllText" method reads the data of a file.

```
string filePath = @"C:\MyData\TestFile.txt";
string testData = File.ReadAllText(filePath);
```

The "ReadAllLines" method reads all the contents of a file and stores each line at a new index in an array of string type.

```
string filePath = @"C:\MyData\TestFile.txt";
string[] testDataLineByLine = File.ReadAllLines(filePath);
```

The "ReadAllBytes" method reads the contents of a file as binary data and stores the data in a byte array.

```
string filePath = @"C:\MyData\TestFile.txt";
byte[] testDataRawBytes = File.ReadAllBytes(filePath);
```

Each of these methods enable the developer to read the contents of a file and load into memory. The ReadAllText method will enable the developer to cache the entire file in memory via a single operation. Whereas the ReadAllLines method will read line-by-line into an array.

Writing Data to Files : data is written to a file. By default, all existing contents are removed from the file, and new content is written.

The "WriteAllText" method enables the developers to write the contents of string variable into a file. If the file exists, its contents will be overwritten. The following code example shows how to write the contents of a string named settings to a new file named settings.txt.

```
string filePath = @"C:\MyData\TestFile.txt";
string data = "C# Corner MVP & Microsoft MVP;";
File.WriteAllText(filePath, data);
```

The "WriteAllLines" method enables the developers to write the contents of a string array to a file. Each entry in the string array will be a new line in the new file.

```
String filePath = @"C:\MyData\TestFile.txt";
string[] data =  "MCT", "MCPD", "MCTS", "MCSD.NET", "MCAD.NET","CSM"};
File.WriteAllLines(filePath,  data)
```

Appending Data to Files – writing information to a file. The only difference is that the existing data in a file is not overwritten. The new data to be written is added at the end of the file.
The "AppendAllText" method enables the developers to write the contents of a string variable at the end of an existing file.

```
String filePath = @"C:\MyData\TestFile.txt";
string data = "Also Certified from IIT Kharagpur";
File.AppendAllText(filePath, data);
```

The "AppendAllLines" method enables the developers to write the contents of a string array to the end of an existing file.

```
String filePath = @"C:\MyData\TestFile.txt";
string[] otherData = { "Worked with Microsoft", "Lived in USA" };
File.AppendAllLines(filePath, otherData);
```

File IO Steps :

1. Creating a file :

We use theCreate() method of theFile class to create a new file in C#. For example,

```
//create a file at patName
FileStream fs = File.Create(pathName);
```

Here, theFile class creates a file atpathName.

Note: If the file already exists, the Create() method overwrites the file.

Example:

```
using System;
using System.IO;
class Program
{
   static void Main()
   {
      // path of the file that we want to create
      string pathName = @"C:\Program\myFile.txt";

      // Create() creates a file at pathName
      FileStream fs = File.Create(pathName);

      // check if myFile.txt file is created at the specified path
      if (File.Exists(pathName))
      {
         Console.WriteLine("File is created.");
      }
      else
      {
         Console.WriteLine("File is not created.");
      }
   }
}
```

2. Open a file :

We use the Open() method of the File class to open an existing file in C#. The method opens aFileStream on the specified file.

Example:

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

```
        string pathName = @"C:\Program\myFile.txt";

       // open a file at pathName
        FileStream fs = File.Open(pathName, FileMode.Open);
    }
}
```

In the above example, notice the code,

```
//opens the file at pathName
FileStream fs = File.Open(pathName,FileMode.Open);
```

Here, theOpen() method opensmyFile.txt file. Here,FileMode.Openspecifies -**open the existing file**.

3. Write to a file :

We use theWriteAllText() method of theFile class to write to a file. The method creates a new file and writes content to that file

Example:

```
using System;
using System.IO;
class Program
{
    static void Main()
    {
        string pathName = @"C:\Program\myFile.txt";

        // create a file at pathName and write "Hello World" to the file
        File.WriteAllText(pathName, "Hello World");
    }
}
```

In the above example, notice the code,
File.WriteAllText(pathName,”Hello World”);

Here,the WriteAllText() method creates myFile.txt at c:\Program directory and writes “Hello Word” to the file.

4. Read a file :

We use theReadAllText() method of the File class to read contents of the file. The method returns a string containing all the text in the specified file.
Let's read the content of the filemyFile.txt where we had written"Hello World".

Exam
ple:

```
using
Syste
m;
using
System.IO;
class
Program
{
    static void Main()
    {
        string pathName = @"C:\Program\myFile.txt";

        // read the content of myFile.txt file
        string readText = File.ReadAllText(pathName);

        Console.WriteLine(readText);
    }
}
```

In the above example, notice the code,

```
//read the content of myFile.txt file
string readText= File.ReadAllText(pathName);
```

The readAllText() method read the file myFile.txt and returns “Hello World”.

Serialization :

serialization is the process of converting object into byte stream so that it can be saved to memory, file or database