

# File Handling in C#

---

SESSION 10

# Objectives

---

*Discuss different classes within System.IO namespace*

*Discuss different kinds of stream handling with C#*

*List various methods and properties used for file Input/Output*

*Implement file handling and other stream input handling with C#*

# IO Namespace and its Classes

---

IO namespace includes classes that facilitate reading and writing of data to data streams and files

Classes of IO namespace used for handling files are:

<b>BinaryReader</b>	<b>TextWriter</b>
<b>BinaryWriter</b>	<b>Directory</b>
<b>Stream</b>	<b>File</b>
<b>TextReader</b>	<b>FileSystemInfo</b>

# BinaryReader and BinaryWriter

---

These classes are derived from System.Object class

These classes are used to format binary data

Data can be read and written from any C# variable to the specified stream

# BinaryReader Class

---

Used for reading binary data

➤ Methods supported are:

Methods	Description
Close()	Used to close the current stream from which data is being read
Read()	Employed to read characters from the specified stream
ReadDecimal()	Reads a decimal value from the specified stream
ReadByte()	Reads a byte value from the specified stream. The position in the stream is advanced by one byte

# BinaryWriter Class

---

It is used for writing binary data from a C# variable to a specified stream

The most commonly used methods of this class are **Close()** and **Write()** methods

The Close() method is similar to the BinaryReader class Close() method

Close() method is used to close the current stream to which the binary data is being written and also the current BinaryWriter

# BinaryReader and BinaryWriter - Example

---

```
using System;
using System.IO;
class BinaryDemo
{
    private const string fname = "Binary.data";
    public static void Main(String[] args)
    {
        //check if file already exists
        if (File.Exists(fname))
        {
            Console.WriteLine("{0} already exists!", fname);
            return;
        }
        // if not existing then create a new empty data file.
        FileStream fs = new FileStream(fname, FileMode.CreateNew);
```

# BinaryReader and BinaryWriter - Example

---

```
// Create the writer for data.
    BinaryWriter w = new BinaryWriter(fs);
    for (int i = 0; i < 11; i++)
    {
        w.Write( (int) i);
    }
Console.WriteLine ("Data has been written to the file!");
    w.Close();
    fs.Close();

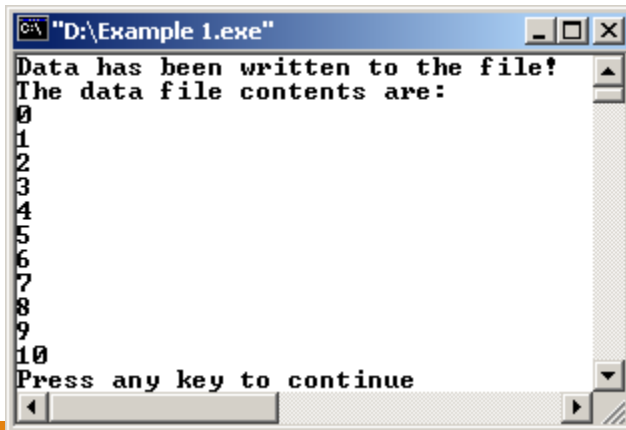
// Create the reader for data.
fs = new FileStream(fname, FileMode.Open, FileAccess.Read);
    BinaryReader r = new BinaryReader(fs);
    Console.WriteLine("The data file contents are:");
```



# BinaryReader and BinaryWriter – Example

---

```
// Read data from the data file.  
for (int i = 0; i < 11; i++)  
{  
    Console.WriteLine(r.ReadInt32());  
}  
w.Close();  
}  
}
```



# Stream Class

---

It is an abstract class from which different classes are being derived

Some of its derived classes are:

- MemoryStream
- BufferedStream
- FileStream
- NetworkStream
- CryptoStream

# MemoryStream class

---

This class is used to read and write data to memory

Some of the methods of MemoryStream are:

Method	Description
Read()	Used to read the MemoryStream and write the value to the buffer.
ReadByte()	Used to read a byte from the MemoryStream
Write()	Used to write values from the buffer into the MemoryStream
WriteByte()	Used to write a byte to the MemoryStream from the buffer.
WriteTo()	Used to write contents of one memory stream into another.

# BufferedStream Class

---

It is used to read and write to the buffer

It has two overloaded constructors with following syntax:

```
public BufferedStream(Stream StName);  
//constructor type 1  
  
public BufferedStream(Stream StName, int bsize);  
//constructor type 2
```

# FileStream Class

---

This class is used to perform read and write operations on files

**Read()** and **Write()** methods are applied for synchronous read and write operations

**BeginRead()** and **BeginWrite()** methods are used for asynchronous read and write operations

The default mode in the FileStream class is synchronous read/write operations

# FileStream Class Constructors

---

Constructors	Description
FileStream(string FilePath, FileMode)	Takes in the path of the file to be read from or written to and any one of the FileMode enumerator values as its arguments.
FileStream(string FilePath, FileMode, FileAccess)	Takes in the path of the file to be read from or written to, any one of the FileMode enumerator values and FileAccess enumerator values as its arguments.
FileStream(string FilePath, FileMode, FileAccess, FileShare)	Takes in the path of the file to be read from or written to, any one of the FileMode enumerator values, FileAccess enumerator values and any one of the FileShare enumerator values as its arguments.

# Enumerators used with FileStream class

---

## **FileMode Enumerators**

- Append
- Create
- CreateNew
- Open
- OpenOrCreate
- Truncate

## **FileAccess Enumerators**

- Read
- Write
- ReadWrite

## **FileShare Enumerators**

- None
- Read
- Write
- ReadWrite

# FileStream Class Example

---

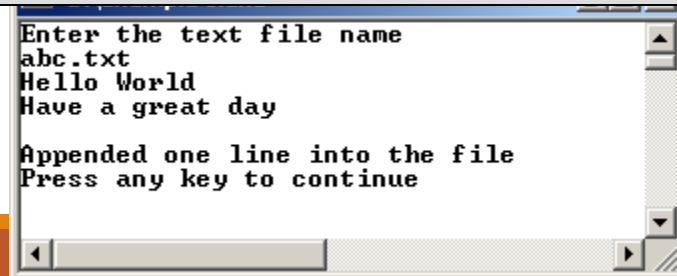
```
using System;
using System.IO;
using System.Text;
class FileStreamDemo
{
    public static void Main()
    {
        Console.WriteLine ("Enter the text file name");
        string fname = Console.ReadLine();
        StreamReader sr = new StreamReader(fname) ;
        string line;
        while ((line = sr.ReadLine()) != null)
        {
            Console.WriteLine (line);
        }
        Console.WriteLine("");
    }
}
```



# FileStream Class - Output

---

```
sr.Close();  
    FileStream filestr = new FileStream(fname,  
    FileMode.Append, FileAccess.Write, FileShare.Write);  
    filestr.Close();  
    StreamWriter sw = new StreamWriter (fname, true,  
    Encoding.ASCII);  
    string NextLine = "This is the appended line.";  
    sw.Write(NextLine);  
    sw.Close();  
    Console.WriteLine ("Appended one line into the file");  
    }  
}
```



# NetworkStream class

---

It is used to send and receive data across the network

It resides within the **System.Net.Sockets** namespace

The Read(), ReadBytes(), Write() and WriteBytes() methods are employed for reading and writing to streams and buffers on the network

# CryptoStream Class

---

It is used to link the stream of data to any cryptography object for the purpose of encrypting the data

It resides within the **System.Security.Cryptography** namespace

# Directory and File Classes

---

The directory class contains static methods that help in handling directories and subdirectories

The static methods of this class can be called without an instance of a directory

The file class contains static methods that help in handling files

It also helps in the creation of FileStream class

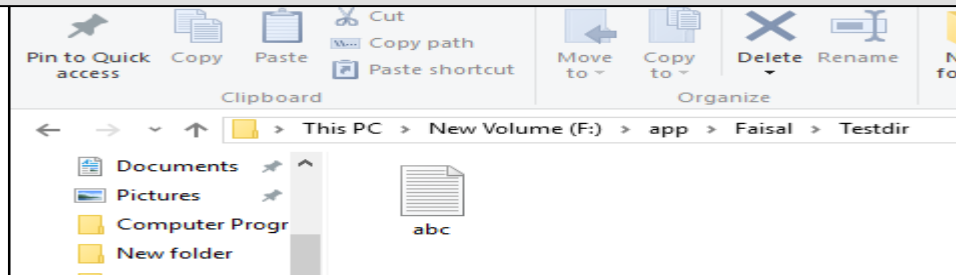
# Methods of the File class

---

Methods	Description
Copy(string SourceFilePath, string DestinationFilePath);	Used to copy the contents of a source file to a destination file in the specified path. If the destination file does not exist, a new file is created with the specified name in the specified path.
Create(string FilePath)	Used to create a file with the specified name in the specified path.
Delete(string FilePath)	Used to delete a file from a specified path
Exists(string FilePath)	Used to verify whether a file with the specified name exists in the specified path. It returns a Boolean value.
Move(string SourceFilePath, string DestinationFilePath)	Used to move the specified file from the source location to the destination location.

# Example of Directory and File Classes

```
using System;
using System.IO;
class DirectoryDemo
{
    static void Main(string[] args)
    {
        Directory.CreateDirectory ("Testdir");
        File.Copy ("D:\\abc.txt", "Testdir\\abc.txt");
        Console.WriteLine("File Content Copied");
    }
}
```



# FileSystemInfo class

---

It is an abstract class from which the **FileInfo** and **DirectoryInfo** classes have been derived

The **DirectoryInfo** class contains methods that can be used to handle directories and subdirectories

The DirectoryInfo class exposes instance methods

The methods of DirectoryInfo class can be called only by an instance of the DirectoryInfo class

The **FileInfo** class contains methods that can be used to handle files

# Properties and Methods of DirectoryInfo Class

---

Properties	Description
FullName	Retrieves the complete path of the directory
Parent	Retrieves the immediate parent directory of the specified subdirectory.
Root	Retrieves the root node of the given path
Methods	Description
Create()	Used to create a directory
CreateSubdirectory(string directorypath)	Creates a subdirectory under the specified directory in the specified path.
MoveTo(string destinationpath)	Moves the current directory to the given destination path.



# Properties and Methods of FileInfo Class

---

Properties	Description
DirectoryName	Contains the full path of the file.
Extension	Used to retrieve the extension of the specified file with the period (.).
Methods	Description
CopyTo(string destinationfile)	Used to copy the contents of the existing file into a new file.
Create()	Used to create a file.
Delete()	Used to permanently delete a file
OpenWrite()	Creates an instance of the FileStream class for, both, read and write operations
OpenRead()	Creates an instance of the FileStream class for read only operation.

# TextReader class

---

It is an abstract base class for the **StreamReader** and **StringReader** classes

These classes can be used to read a sequential series of characters

The StreamReader reads a character in a byte stream and converts it to the specified encoding

The StringReader class is used to read data from an input string

# Methods of StreamReader class

---

Methods	Description
Read()	Used to read a character from the byte stream and move the current position to the next character.
ReadLine()	Reads a sequence of, or a line of character from the byte stream and outputs a string data.
ReadToEnd()	Used to read the byte stream from the current position in the stream to the end of the stream.

# Example of StreamReader class

```
using System;
using System.IO;
public class TextDemo
{
    static string ans="y";
    public static void Main(String[] args)
    {
        Console.WriteLine("1. Read File ");
        Console.WriteLine("2. Read String ");
        Reading();
    }
    static void Reading()
    {
        try
        {
            if(ans=="y" || ans=="Y")
            {
```

## Example of StreamReader class Contd...

---

```
Console.Write ("Enter Your Choice [1/2]: ");
int choice=Convert.ToInt32(Console.ReadLine());
if (choice==1)
{
    Console.WriteLine ("Enter the file name: ");
    string Filename = Console.ReadLine();
    if (!File.Exists(Filename))
    {
        Console.WriteLine("{0} does not exist!" ,Filename);
        return;
    }
    StreamReader sr = File.OpenText(Filename);
    String input;
    Console.WriteLine("The contents of the file are: \n");
```

## Example of StreamReader class Contd...

---

```
while ((input= sr.ReadLine())!=null)
{
    Console.WriteLine (input);
}
Console.WriteLine ("The end of the stream is reached.");
sr.Close();
    Console.Write( "Do you want to continue [Y/N]:");
    ans= Console.ReadLine();
    Reading();
}
else if (choice==2)
{
    Console.Write ("Enter a string: ");
    String str = Console.ReadLine();
    char[] b = new char [str.Length];
```

# Example of StreamReader class – Contd...

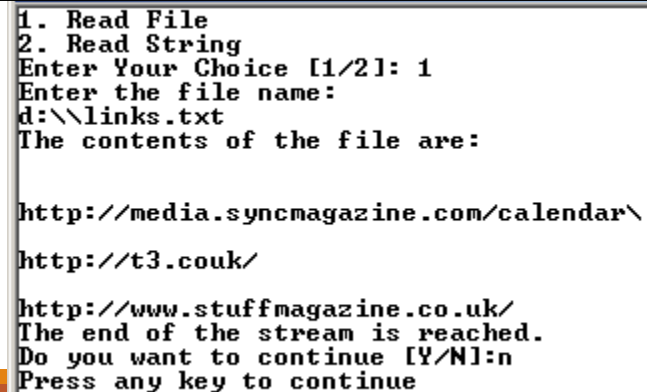
---

```
StringReader sr = new StringReader (str);
    sr.Read(b, 0, str.Length);
    Console.WriteLine (b);
    Console.Write ("Do you want to continue [Y/N]:");
    ans= Console.ReadLine();
    Reading();
}
else
{
    Console.WriteLine ("Enter either 1 or 2 as your
choice");
}
}
}
```

# Example of StreamReader class – Contd...

---

```
catch (Exception e)
{
    Console.WriteLine (e.StackTrace);
    Console.WriteLine (e.Message);
}
}
```



1. Read File  
2. Read String  
Enter Your Choice [1/2]: 1  
Enter the file name:  
d:\\links.txt  
The contents of the file are:  
  
http://media.syncmagazine.com/calendar\  
  
http://t3.couk/  
  
http://www.stuffmagazine.co.uk/  
The end of the stream is reached.  
Do you want to continue [Y/N]:n  
Press any key to continue



# TextWriter class

---

It is an abstract base class for classes that can be used to write sequential characters

The **StreamWriter** and **StringWriter** classes are two of the derived classes of the TextWriter class

The StreamWriter writes characters to a stream in a specified encoding

The StringWriter class is used to write data to a string

Methods of StreamWriter class

- Write()
- WriteLine()

# Methods of StreamWriter

---

Method	Description
Write()	Used to write a character from the stream and move the current position to the next character.
WriteLine()	Writes a sequence of a line of characters to the stream. It adds a line terminator to mark the end of the string.

# Example of StreamWriter class

---

```
using System;
using System.IO;
using System.Text;
public class Writer
{
    static string ans="y";
    public static void Main(String[] args)
    {
        Writing();
    }
    static void Writing()
    {
        if (ans=="y" || ans=="Y")
        {
            Console.Write ("Enter the file name: ");
            string Filename = Console.ReadLine();
        }
    }
}
```

# Output - StreamWriter

---

```
if (!File.Exists(Filename))
{
    Console.WriteLine("{0} does not exist!",Filename);
    return;
}
StreamWriter sr = File.AppendText(Filename);
Console.Write ("Enter a string to be
                written to the file: ");

String str = Console.ReadLine();
sr.WriteLine(str);
sr.Close();
Console.Write ("Do you want to continue [Y/N]: ");
ans= Console.ReadLine();
Writing();
    }
}
}
```