Los programas muy a menudo necesitan enviar datos a un determinado destino, o bien leerlos de una determinada fuente externa, como por ejemplo puede ser un fichero para almacenar datos de forma permanente, o bien enviar datos a través de la red, a memoria, o a otros programas. Esta entrada/salida de datos se realiza por medio de ***flujos (streams) de datos***, a través de los cuales un programa puede recibir o enviar datos en serie.

Si queremos transferir estructuras de datos complejas, deberemos convertir estas estructuras en secuencias de bytes que puedan ser enviadas a través de un flujo. Esto es lo que se conoce como ***serialización.*** Comenzaremos viendo los fundamentos de los flujos de entrada y salida en .Net, para a continuación pasar a estudiar los flujos que nos permitirán serializar diferentes tipos de datos de forma sencilla.

.Net implementa una serie de clases, llamadas ***flujos (Streams),*** que son las encargadas de comunicarse con los dispositivos de almacenamiento. El funcionamiento de estos flujos, desde el punto de vista del programador, no depende del dispositivo hardware con el que está asociado, lo que nos liberará del trabajo que supone tener en cuenta las características de cada uno.

Los flujos pueden ser de entrada o de salida, según que sean para guardar o recuperar información.
Por otra parte, atendiendo al tipo de datos que se transmiten, los flujos son de dos tipos:

- Byte, si transmiten bytes, enteros entre 0 y 255. Esto, en realidad, permite manipular cualquier tipo de datos.
- Carácter, si se asocia a archivos u otras fuentes de tipo texto.

Acceder a un archivo usando StreamWriter y ReadWriter

Utilizaremos la clase StreamWriter para escribir en archivos y la clase StreamReader para leer desde archivos. Estas clases están orientadas a obtener **caracteres** como salida a diferencia de las clases que heredan de Stream que están orientadas a obtener **bytes**.

StreamWriter

El proceso típico de escritura de datos mediante un **StreamWriter** comprende los siguientes pasos:

- **Instanciar** un objeto de esta clase mediante alguno de los constructores disponibles.
  Aquí creamos un nuevo archivo para escribir datos sobre él, o abrimos uno existente.
- **Escritura de texto** mediante los métodos **WriteLine( )** y **Write( )**.
  - *WriteLine() escribe el texto pasado como parámetro, y añade los caracteres especiales de retorno de carro y nueva línea.*
  - *Write() escribe el texto pasado y deja el puntero de escritura a partir del último carácter escrito, con lo que no produce un cambio automático de línea. Deberemos utilizar la propiedad **NewLine** para introducir manualmente un salto de línea.*
- **Cierre del Stream** con el método **Close( )**.

Ejemplo 1: Proceso típico

```
Imports System.IO
Public Class Form1

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e
      As System.EventArgs) Handles Button1.Click

        ' creamos un stream de escritura, y al mismo tiempo un
        ' nuevo archivo para escribir texto sobre él
        Dim fichero As StreamWriter
        fichero = New StreamWriter("NOTAS.txt")

        ' escribir líneas
        fichero.WriteLine("esta es la primera línea")
        fichero.WriteLine("segunda línea de texto")

        ' ahora escribimos texto pero sin provocar un salto de línea
        fichero.Write("Juan y Luna ")
        fichero.Write("van de paseo")
        fichero.Write(fichero.NewLine)     ' esto introduce el salto de
      línea
        fichero.WriteLine("con esta línea cerramos")

        ' cerrar el stream y el archivo asociado
        fichero.Close()
        MessageBox.Show("Fichero creado")

    End Sub
End Class
```

La diferencia entre el método Write y WriteLine, es que el segundo inserta un salto de línea al final de los datos ingresados, haciendo que la próxima vez que se quiera insertar, se hará en la siguiente línea.

Ejemplo 2. ¿y si ya existe el archivo?

En el caso de que el archivo sobre el que vamos a escribir ya exista, podemos utilizar un constructor de StreamWriter que nos permite especificar si vamos a añadir texto al archivo o vamos a sobrescribirlo perdiendo el texto que antes contuviera:

```
'abre el archivo y se sitúa al final del texto para añadir
fichero = New StreamWriter("NOTAS.txt", True)

'se elimina el contenido previo del archivo
fichero = New StreamWriter("NOTAS.txt", False)
```

Ejemplo 3. Otros constructores

La clase StreamWriter también tiene otro constructor donde se le puede pasar un Stream que tengamos creado, como fs en el siguiente ejemplo, en lugar de la ruta del archivo.

```
Dim fs As FileStream
fs = New FileStream("NOTAS.txt", FileMode.Create, FileAccess.Write,
   FileShare.None)
Dim fichero As StreamWriter
fichero = New StreamWriter(fs)
```

El uso de este constructor especifica que hay un canal o intermediario (el FileStream) entre nuestro archivo y el programa, y un objeto (el StreamWriter) que escribe sobre dicho canal.

El constructor utilizado en los ejemplos anteriores hace lo mismo aunque internamente. La clase StreamWriter en su interior, se encarga de crear y utilizar el Stream necesario para apuntar al archivo de texto representado por la ruta que se ha utilizado en el constructor del ejemplo, esto a un nivel de muchas peticiones puede ser perjudicial para el performance.

StreamReader

Un objeto StreamReader realiza operaciones de lectura de texto sobre un archivo.
Proceso típico:

- **instanciar** el objeto con uno de sus constructores, abriendo un archivo para leer;
- ejecutar alguno de los **métodos de lectura** del StreamReader,
- y cerrar el objeto con **Close( )**

Métodos de lectura más comunes de este objeto:

- **ReadLine()** devuelve una línea del archivo;
- **ReadToEnd()** devuelve el resto del contenido del archivo desde el punto en el que se encontrara el Stream al realizar la última lectura.

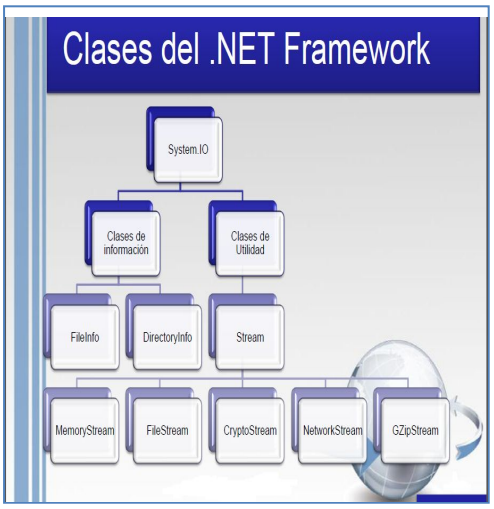
Ejemplo 4:

```
Imports System.IO
Public Class Form1

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
Dim Lector As StreamReader = New StreamReader("NOTAS.txt")
        'Leer una primera línea
        Dim Linea As String
        Linea = Lector.ReadLine()
        MessageBox.Show("Contenido de la primera linea: " & Linea)


        'Leer el resto del fichero
        Dim Texto As String
        Texto = Lector.ReadToEnd()
        MessageBox.Show("Contenido del resto del archivo: " & Texto)
       Lector.Close()
End Sub
```


Archivos VS Flujos
• Un archivo es una colección de datos almacenado en un disco con un cierto nombre.
• Un flujo (stream) es con lo que se pueden llevar a cabo operaciones de lectura y escritura.


Clases del .NET Framework
System.IO
Clases de información
Clases de Utilidad
FileInfo
DirectoryInfo
Stream
MemoryStream
FileStream
CryptoStream
NetworkStream
GZipStream


Clases más importantes (System.IO)
• StreamReader
• StreamWriter
• BinaryReader
• BinaryWriter
• XmlTextReader
• XmlTextWriter
• File
• FileStream
• Stream