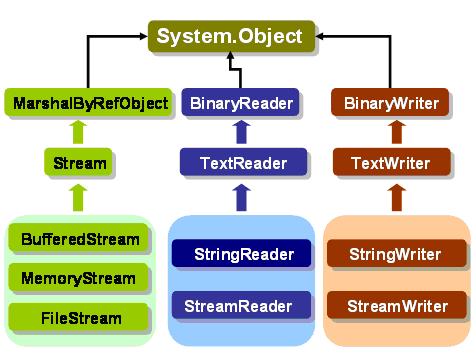
La lectura y escritura a un archivo son hechas usando un concepto genérico llamado **stream,** transferencia de un punto al otro como un flujo de datos.

En .NET puede encontrar muchas clases que trabaja con archivos o con datos de memoria

  
**Figura 1.** *Clases del Framework .NET para el uso de Streams.*

**Principales operaciones de stream**

Los streams permiten realizar tres operaciones principales:

1. Podemos leer de streams.

La lectura es la transferencia de datos desde un stream a una estructura de datos (como una matriz de bytes).

2. Podemos escribir a streams.

La escritura es la transferencia de datos desde una estructura de datos a un stream.

3. Los streams pueden soportar búsqueda.

La búsqueda es la consulta y modificación de la posición actual en un stream. La capacidad de búsqueda depende del tipo de repositorio de seguridad que tenga un stream. Por ejemplo: el concepto de posición actual no se aplica a streams de red y, por tanto, típicamente los streams de red no soportan búsqueda.

**Leyendo desde un archivo de texto**

using System;

using System.IO;

static void Main(string[] args)

{

string fileName = "temp.txt";

FileStream stream = new FileStream(fileName, FileMode.Open, FileAccess.Read);

StreamReader reader = new StreamReader(stream);

while (reader.Peek() > -1) Console.WriteLine(reader.ReadLine());

reader.Close();

}

**Escribiendo en un archive de texto**

using System;

using System.IO;

static void Main(string[] args)

{

string fileName = "temp.txt";

FileStream stream = new FileStream(fileName, FileMode.OpenOrCreate, FileAccess.Write);

StreamWriter writer = new StreamWriter(stream);

writer.WriteLine("Esta es la primera línea del archivo.");

writer.Close();

}

**Creando un archivo y escribiendo en este**

Este ejemplo usa el método CreateText() el cual crea un Nuevo archive y retorna un objeto StreamWriter que escribe a un archivo usando formato UTF-8.

using System;

using System.IO;

static void Main(string[] args)

{

string fileName = "temp.txt";

StreamWriter writer = File.CreateText(fileName);

writer.WriteLine("Este es mi Nuevo archivo creado.");

writer.Close();

}

**Insertando texto en un archivo**

using System;

using System.IO;

static void Main(string[] args)

{

try

{

string fileName = "temp.txt";

// esto inserta texto en un archivo existente, si el archivo no existe lo crea

StreamWriter writer = File.AppendText(fileName);

writer.WriteLine("Este es el texto adicionado.");

writer.Close();

}

catch

{

Console.WriteLine("Error");

}

}

**Leyendo un archivo binario**

using System;

using System.IO;

static void Main(string[] args)

{

try

{

string fileName = "temp.txt";

int letter = 0;

FileStream stream = new FileStream(fileName, FileMode.Open, FileAccess.Read);

BinaryReader reader = new BinaryReader(stream);

while (letter != -1)

{

letter = reader.Read();

if (letter != -1) Console.Write((char)letter);

}

reader.Close();

stream.Close();

}

catch

{

Console.WriteLine("Error");

}

}

**Escribiendo en un archivo binario**

static void Main(string[] args)

{

try

{

string fileName = "temp.txt";

// data a ser guardada

int[] data = {0, 1, 2, 3, 4, 5};

FileStream stream = new FileStream(fileName, FileMode.Open, FileAccess.Write);

BinaryWriter writer = new BinaryWriter(stream);

for(int i=0; i<data.Length; i++)

{

// números son guardados en formáto UTF-8 format (4 bytes)

writer.Write(data[i]);

}

writer.Close();

stream.Close();

}

catch

{

Console.WriteLine("Error");

}

**Ver cambios en el file system**

.NET Framework prove la calse System.IO.FileSystemWatcher la cual permite ver si hay cambios en el file system.

using System;

using System.IO;

class WatcherSample

{

static void Main(string[] args)

{

// ver los cambios en el directorio de la aplicación y sobre todos los archivos

FileSystemWatcher watcher = new FileSystemWatcher(System.Windows.Forms.Application.StartupPath, "\*.\*");

// ver el nombre del archivo y tamaño cambiado

watcher.NotifyFilter = NotifyFilters.FileName | NotifyFilters.Size;

watcher.Changed += new FileSystemEventHandler(OnChange);

watcher.Created += new FileSystemEventHandler(OnChange);

watcher.Deleted += new FileSystemEventHandler(OnChange);

watcher.Renamed += new RenamedEventHandler(OnChange);

watcher.EnableRaisingEvents = true;

// espera por una tecla para terminar la aplicación

Console.ReadLine();

}

private static void OnChange(object sender, FileSystemEventArgs e)

{

Console.WriteLine("File: {0} - change type: {1}", e.FullPath, e.ChangeType);

}

private static void OnChange(object sender, RenamedEventArgs e)

{

Console.WriteLine("File: {0} renamed to {1}", e.OldName, e.Name);

}

}

#### **Las clases Stream proporcionadas por el .NET Framework**

El .NET Framework contiene varias clases stream que derivan de la clase **System.IO.Stream**. El espacio de nombres **System.Net.Sockets** contiene la clase **NetworkStream**. **NetworkStream** proporciona el stream subyacente de datos para el acceso a redes.

El espacio de nombres **System.IO** contiene las clases **BufferedStream**, **MemoryStream** y **FileStream**, derivadas de la clase **System.IO.Stream**.

##### **Clase BufferedStream**

La clase **BufferedStream** se utiliza para invocar el proceso de lectura de buffer desde otro stream, y escritura de buffer a otro stream. Un buffer es un bloque de bytes en memoria que se utiliza para almacenar datos en caché, reduciendo así el número de llamadas al sistema operativo. Los buffers pueden utilizarse para mejorar el rendimiento de la lectura y escritura. Ninguna otra clase puede heredar de la clase **BufferedStream**.

##### **Clase MemoryStream**

La clase **MemoryStream** proporciona un método de creación de streams que utiliza la memoria (en lugar de un disquete o una conexión a red) como repositorio de seguridad. La clase **MemoryStream** crea un stream desde una matriz de bytes.

##### **Clase FileStream**

La clase **FileStream** se utiliza tanto para leer de archivos como para escribir a ellos. De forma predeterminada, la clase **FileStream** abre archivos síncronamente, pero también proporciona un constructor para abrir archivos asíncronamente.

|  |  |
| --- | --- |
| La siguiente tabla describe algunas clases reader y writer utilizadas habitualmente. **Clase** | **Descripción** |
| **BinaryReader** y **BinaryWriter** | Estas clases leen y escriben tipos como valores binarios en una codificación específica a y desde un stream. |
| **TextReader** y **TextWriter** | Las implementaciones de estas clases están diseñadas para introducir y extraer caracteres. |
| **StreamReader** y **StreamWriter** | Estas clases se derivan de las clases **TextReader**y **TextWriter**, que leen sus caracteres desde un stream y escriben sus caracteres a un stream respectivamente. |
| **StringReader** y **StringWriter** | Estas clases también derivan de las clases **TextReader** y **TextWriter**, pero leen sus caracteres desde una cadena y escriben sus caracteres a una clase **StringBuilder** respectivamente. |