

Потоци



Учителски екип

Обучение за ИТ кариера

<https://it-kariera.mon.bg/e-learning/>

Съдържание

1. Какво са потоците?
2. Reader и Writer класове
3. Типове потоци
 - File, Memory, Network потоци
 - Crypto, Gzip потоци



Какво е поток?

- Потоците са създадени за **пренос** (четене и запис) на данни
 - Представяват **подредена последователност от байтове**
 - Осигуряват последователен достъп до своите елементи
 - Трябва **да се отворят** преди употреба и **да се затворят** накрая
- Има **различни типове потоци** за разните типове данни:
 - За достъп до файлове и мрежа, потоци в паметта и други



Поток - пример

Length = 9

F

i

l

e

s

a

n

d

46

69

6c

65

73

20

61

6e

64

Position



Buffer

46	69	6c	65	73	20	61	6e	64	
----	----	----	----	----	----	----	----	----	--

- **Position** - текущата позиция в потока
- **Buffer** - пази **n** байта от потока от **текущата позиция**

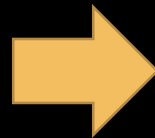
Reader и Writer класове

- Reader и writer са класове, улесняващи работата с потоците
- Има два типа потоци:
 - Текстово четене/запис – **StreamReader / StreamWriter**
 - Имат методи **.ReadLine()**, **.WriteLine()** (като **Console.***)
 - Двоично четене/запис – **BinaryReader / BinaryWriter**
 - Имат методи за работа с примитивни типове – **.ReadInt32()**, **.ReadBoolean()**, **WriteChar()** и т.н.

Задача: Четене на файл

- Прочетете цялото съдържание на **Program.cs** файла
- Отпечатайте го на конзолата с номера на редове

```
using System;  
using System.IO;  
  
class Program  
{
```



```
Line 1: using System;  
Line 2: using System.IO;  
Line 3:  
Line 4: class Program  
Line 5: {
```

Решение: Четене на файл

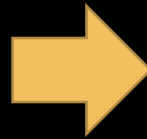
```
StreamReader reader = new StreamReader("somefile.txt");
using (reader)
{
    int lineNumber = 0;
    string line = reader.ReadLine();
    while (line != null)
    {
        lineNumber++;
        Console.WriteLine("Line {0}: {1}", lineNumber, line);
        line = reader.ReadLine();
    }
}
```

За затваряне на потока и освобождаване на буферите накрая

Задача: Запис във файл

- Прочетете вашия Program.cs файл
- Обърнете наобратно всеки негов ред
- Запишете резултата в reversed.txt

```
using System.IO;  
  
class Program  
{
```



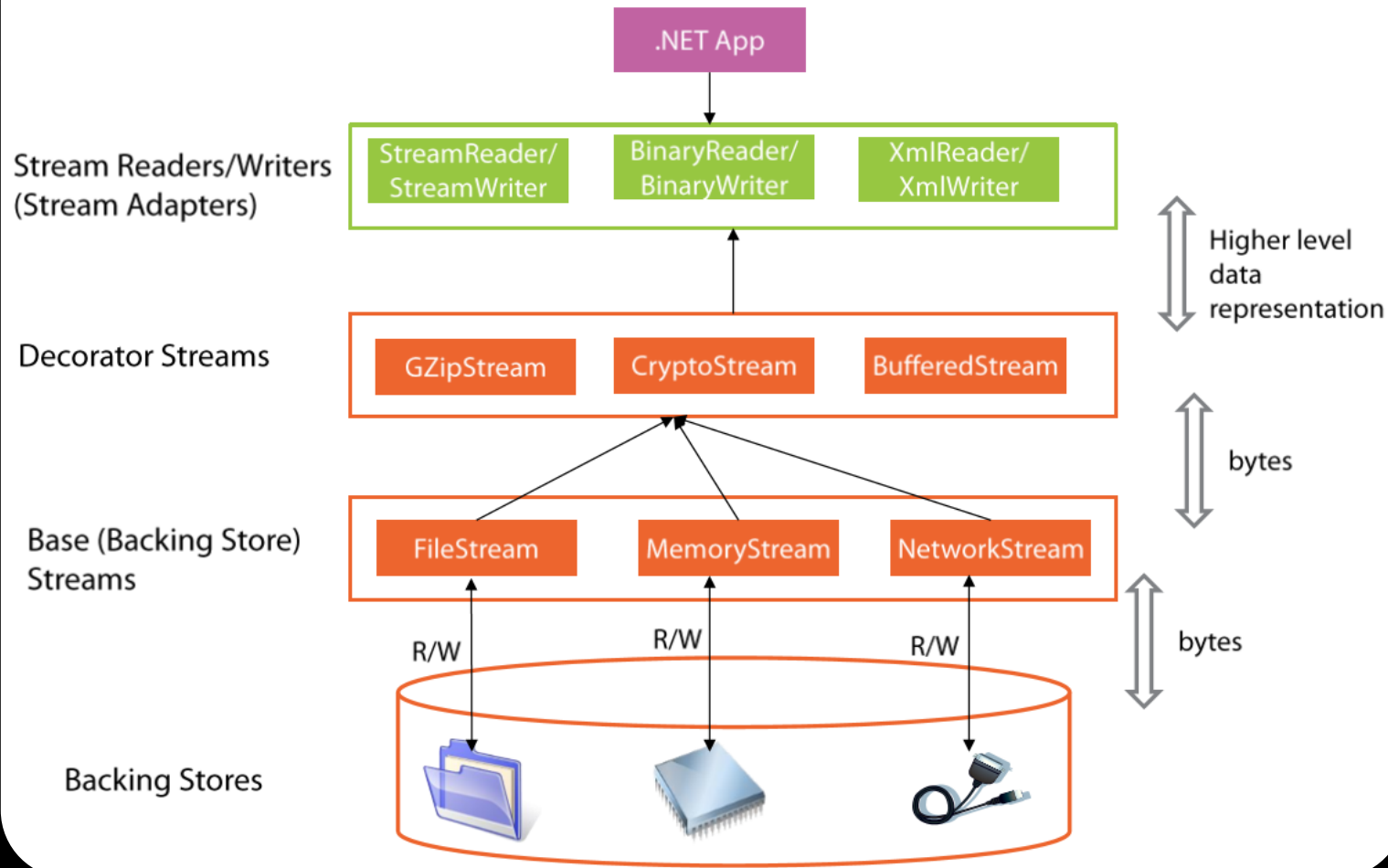
```
;OI.metsyS gnisu  
  
margorP ssalc  
{
```


Запис на текста на файл наобратно - пример

```
using (var reader = new StreamReader("../..//Program.cs"))
{
    using (var writer = new StreamWriter("../..//reversed.txt"))
    {
        string line = reader.ReadLine();
        while (line != null)
        {
            for (int i = line.Length - 1; i >= 0; i--)
            {
                writer.Write(line[i]);
            }
            writer.WriteLine();
            line = reader.ReadLine();
        }
    }
}
```

Типове потоци в .NET

The Overall Architecture



Класът `System.IO.Stream`

- Базовият клас за всички потоци е `System.IO.Stream`
- Той има **дефинирани методи** за основните операции с потоци
- Някои потоци не поддържат **четене, запис или позициониране**
 - Затова има свойства `CanRead`, `CanWrite` и `CanSeek`
 - Потоците, които поддържат позициониране имат свойства `Position` и `Length`

Методи на класа `System.IO.Stream`

- `int Read(byte[] buffer, int offset, int count)`
 - Чете **count** байта от входящия поток, започвайки от дадена **offset** позиция
 - Връща **броя прочетени байтове** или 0 ако е достигнат края
 - Може да замръзне за неопределено време докато прочете поне 1 байт
 - Може да прочете по-малко от обявения брой байтове

F	i	l	e	s	a	n	d	
46	69	6c	65	73	20	61	6e	64

Методи на класа `System.IO.Stream` (2)

- `Write(byte[] buffer, int offset, int count)`
 - Записва поредица от **count** байта в изходящ поток, започвайки от дадена **offset** позиция
 - Може да замръзне за неопределено време, докато изпрати всички байтове по назначение
- `Flush()`
 - Изпраща вътрешно буферираните данни към тяхното назначение (устройство за съхранение на данни, за вход/изход или друго)

Методи на класа `System.IO.Stream` (3)

- `Close()`
 - Извиква `Flush()`
 - Прекъсва връзката към устройството
 - Освобождава заетите ресурси
- `Seek(offset, SeekOrigin)` – премества позицията (ако това се поддържа като операция) с определено отместване спрямо началото, края или текущата позиция

Класът FileStream

- Наследява класът **Stream** и всичките му методи и свойства
 - Поддържа четене, запис, позициониране и т.н.

```
FileStream fs = new FileStream(string fileName, FileMode  
[, FileAccess [, FileShare]]);
```

Опционални параметри

- **FileMode** – режим на отваряне
 - Open, Append, Create, CreateNew, OpenOrCreate, Truncate
- **FileAccess** – режим на работа с файла: Read, Write, ReadWrite
- **FileShare** – права за достъп за другите потребители докато файлът е отворен: None, Read, Write, ReadWrite

Запис на текст във файл – пример


```
string text = "Кирилица";  
var fileStream = new FileStream("../log.txt",  
                                FileMode.Create);  
try  
{  
    byte[] bytes = Encoding.UTF8.GetBytes(text);  
    fileStream.Write(bytes, 0, bytes.Length);  
}  
finally  
{  
    fileStream.Close();  
}
```

try-finally гарантира, че
потокът винаги ще се затвори

Encoding.UTF8.GetBytes()
връща прилежащите байтове
за символите в текста

Копиране на файл – пример

```
using (var source = new FileStream(SheepImagePath, FileMode.Open))
{
    using (var destination =
        new FileStream(DestinationPath, FileMode.Create))
    {
        while (true)
        {
            int readBytes = source.Read(buffer, 0, buffer.Length);
            if (readBytes == 0)
                break;
            destination.Write(buffer, 0, readBytes);
        }
    }
}
```



using автоматично затваря потока

Четене на низ в паметта – пример

```
string text = "In-memory text.";
byte[] bytes = Encoding.UTF8.GetBytes(text);
using (var memoryStream = new MemoryStream(bytes))
{
    while (true)
    {
        int readByte = memoryStream.ReadByte();
        if (readByte == -1)
            break;
        Console.WriteLine((char) readByte);
    }
}
```


Прост уеб сървър – пример

```
var tcpListener = new TcpListener(IPAddress.Any, PortNumber);
tcpListener.Start();
Console.WriteLine("Listening on port {0}...", PortNumber);

while (true)
{
    using (NetworkStream stream = tcpListener.AcceptTcpClient().GetStream())
    {
        byte[] request = new byte[4096];
        stream.Read(request, 0, 4096);
        Console.WriteLine(Encoding.UTF8.GetString(request));

        string html = string.Format("<html><body><h1>{0}</h1></body></html>", DateTime.Now) +
            " Welcome to my awesome site!</h1></body></html>";
        byte[] htmlBytes = Encoding.UTF8.GetBytes(html);
        stream.Write(htmlBytes, 0, htmlBytes.Length);
    }
}
```

Получава потока

Чете заявката

Записва отговора

Класът `BufferedStream`

- Буферира данните и така увеличава производителността
- Прочитане на дори 1 байт води до прочитане на още килобайти в аванс
 - Потокът ги пази във вътрешен буфер
- Следващото четене връща данни от вътрешния буфер
 - Много бърза операция
- Записаните данни се съхраняват във вътрешния буфер
 - Това е много бърза операция
- Когато буферът се препълни:
 - Се извиква `Flush()` и данните се изпращат по назначение

Други потоци

- .NET поддържа и други специални потоци
 - Те работят като обичайните, но предоставят още функции
 - **CryptoStream** криптира при запис и декриптира при четене
 - **GzipStream** компресира и разкомпресира данните
 - **PipedStream** е за четене/запис на данни към няколко процеса



Обобщение

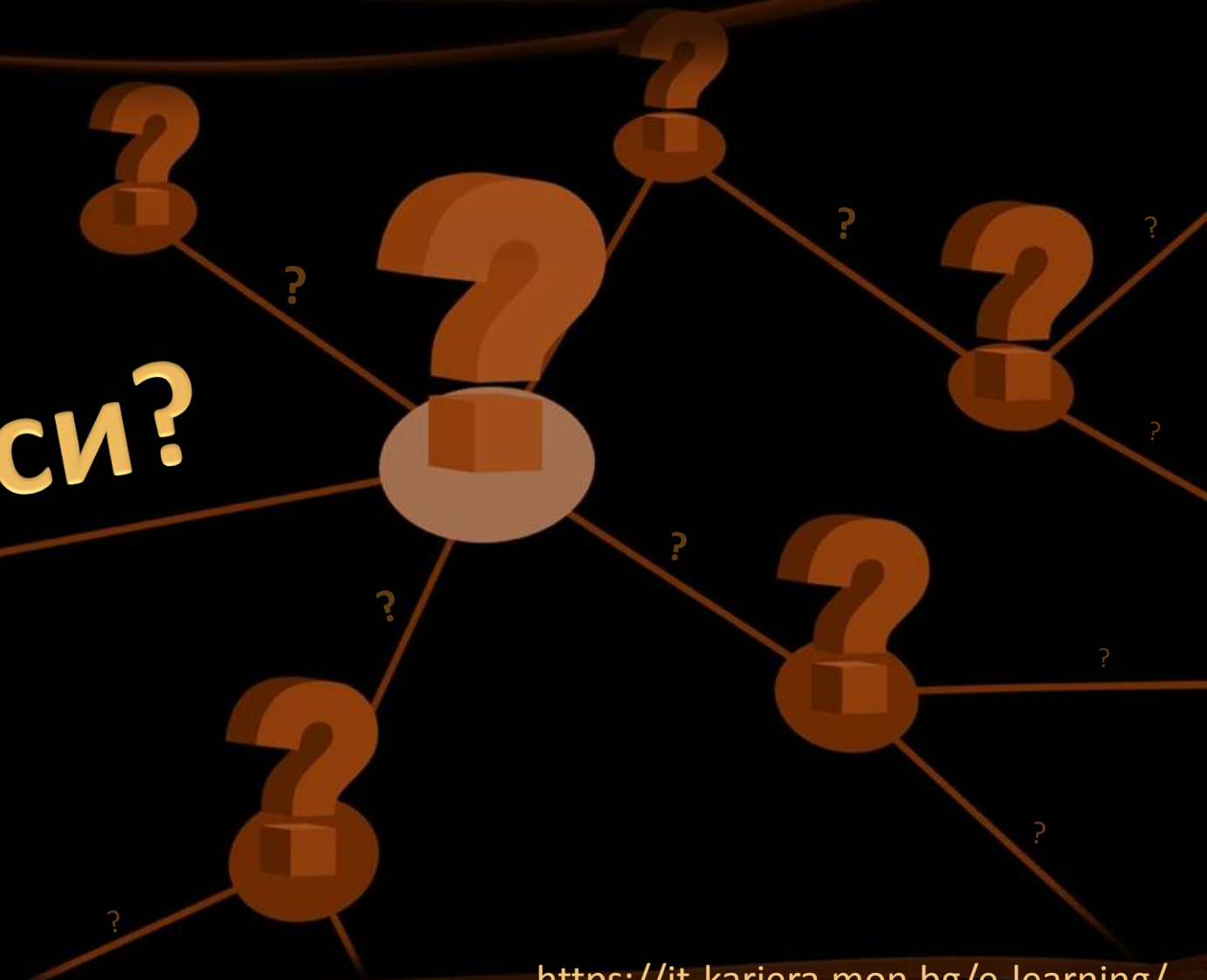
- Потоците са подредени **поредици** от байтове
 - Използват се за входно/изходни операции
 - Могат да са за четене, за запис или и за двете
 - Може да са с различна природа – файл, мрежа, памет, устройство и т.н.
- Reader и writer класовете улесняват работата с потоците чрез предоставяне на допълнителна функционалност (например четене на цели редове)
- Винаги затваряйте потоците чрез **using(...)** или **try-finally**



Потоци

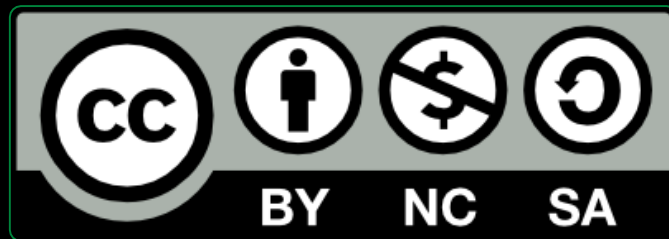


Въпроси?



Лиценз

- Настоящият курс (слайдове, примери, видео, задачи и др.) се разпространяват под свободен лиценз "Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International"



- Благодарности: настоящият материал може да съдържа части от следните източници
 - Книга "Основи на програмирането със C#" от Светлин Наков и колектив с лиценз CC-BY-SA