

# C# Advanced Tekst bestanden (inleiding)

Koen Bloemen



**DE HOGESCHOOL  
MET HET NETWERK**

Elfde-Liniestraat 24, 3500 Hasselt, [www.pxl.be](http://www.pxl.be)





StreamReader  
CSV files  
Fixed-width files

# Streams

- Om bestanden te manipuleren of te lezen gebruiken we de namespace IO (input-output).

```
using System.IO;
```

- StreamReader: lezen van bestand
  - StreamWriter: schrijven naar bestand
- Een bestand wordt gezien als een stream van karakters. Zolang er nog onverwerkte karakters in het bestand zitten, is het einde van de stream nog niet bereikt.
- Een StreamReader kan niet verder lezen dan het einde van de stream.



# Paden bij lezen/schrijven van bestanden

- Standaard relatief lezen/schrijven ten opzichte van programma (de .exe).
  - automatisch in `\bin\debug` bij .NET Framework project
  - bij .NET Core project is dit `\bin\debug\netcoreappX.Y` (vb/ versie 3.1)

```
StreamReader sr = new StreamReader("test.txt");
```

- Relatief pad

```
StreamReader sr = new StreamReader(@"bestanden\tekstbestanden\test.txt");
```

- 1 mapje teruggaan via .. (relatief pad)

```
StreamReader sr = new StreamReader(@"..\test.txt");  
StreamReader sr = new StreamReader("../test.txt"); //indien geen @, doe \\ ipv \
```

- Absoluut pad (best vermijden!)

```
StreamReader sr = new StreamReader(@"C:\some\location\test.txt");
```

# StreamReader: bestand lezen regel per regel

- Voeg de juiste namespace toe

```
using System.IO;
```

- Maak StreamReader aan die bestand kan lezen

```
StreamReader sr = new StreamReader("test.txt");
```

- Lees het bestand nu regel per regel tot we bij het einde gekomen zijn

```
while (!sr.EndOfStream) {  
    string lijn = sr.ReadLine(); // lees nieuwe regel  
    Console.WriteLine(lijn); // print hem af  
}
```

- Bestand sluiten!!!

```
sr.Close();
```

# StreamReader: bestand lezen in 1 keer

- Voeg de juiste namespace toe

```
using System.IO;
```

- Maak StreamReader aan die bestand kan lezen

```
StreamReader sr = new StreamReader("test.txt");
```

- Lees het bestand nu regel per regel tot we bij het einde gekomen zijn

```
string heleBestand = sr.ReadToEnd();  
Console.WriteLine(heleBestand);
```

- Bestand sluiten!!!

```
sr.Close();
```

# CSV files

- CSV = Comma Separated Value
- Velden of waarden worden gescheiden door een scheidingsteken
  - Komma
  - Puntkomma
  - Eender welk teken
- File extensie
  - .txt
  - .csv
- Wordt gebruikt door heel veel applicaties om gegevens te lezen en/of weg te schrijven

# CSV file lezen

- Tekst inlezen en opsplitsen via `String.Split()` volgens scheidingsteken

```
using (StreamReader sr = new StreamReader(@"..\..\KommaBestand.txt"))
{
    // Tekst inlezen regel per regel
    while (!sr.EndOfStream)
    {
        // Splits ingelezen regel op volgens ;
        string[] waarden = sr.ReadLine().Split(',');
        Console.WriteLine($"{waarden[0]} werkt in {waarden[1]}");
    }
}
```

- Denk eraan
  - Dankzij `using () {...}` moeten we geen `sr.Close()` doen!



# Fixed-width text files lezen

- Elke regel heeft vaste breedte (dus een vast aantal karakters).
- Tekst lezen met Substring() en spaties wegdoen met Trim()

```
using (StreamReader sr = new StreamReader(@"..\..\VastBestand.txt"))
{
    while (!sr.EndOfStream)
    {
        string lijn = sr.ReadLine();
        string veld1 = lijn.Substring(0, 19).Trim();
        string veld2 = lijn.Substring(20, 9).Trim();
        string veld3 = lijn.Substring(30, 14).Trim();
        Console.WriteLine($"{veld2} {veld1} werkt in {veld3}");
    }
}
```