C# Advanced
File I/O

Koen Bloemen
Sander De Puydt
Erdem Yaraci

DE HOGESCHOOL MET HET NETWERK

Elfde-Liniestraat 24, 3500 Hasselt
www.pxl.be

Tekstbestanden

Introductie

Relatieve paden

StreamWriter

StreamReader

FileStream

File class

FileInfo class

Introductie

- Wanneer je bestanden wil maken, aanpassen of uitlezen, dan werk je met streams.
- Stream is een reeks van bytes
- 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
  - FileStream: meer opties meegeven aan StreamReader/StreamWriter
  - File: static class met hulpmethods
  - FileInfo: object met hulpmethods

Relatieve paden

- 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)

```
StreamWriter sw = new StreamWriter("test.txt");
```

- Relatief pad

```
StreamWriter sw = new StreamWriter(@"bestanden\tekstbestanden\test.txt");
```

- 1 mapje teruggaan via .. (relatief pad)

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

- Absoluut pad (best vermijden!)

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

StreamWriter: schrijven naar een bestand

- Voeg de juiste namespace toe

```
using System.IO;
```

- Tekstbestand aanmaken en openen

```
StreamWriter sw = new StreamWriter("test.txt");
```

- Naar tekstbestand schrijven

```
sw.WriteLine("Volgorde van getallen:");
for (int i = 0; i < 10; i++)
  sw.Write($"{i} ");
sw.WriteLine(); // lege regel
```

- Bestand sluiten!!!

```
sw.Close();
```

StreamWriter: opdracht

- Schrijf een console programma dat regels tekst wegschrijft in een bestand die worden ingegeven tot de gebruiker de regel “exit” doorgeeft.
- Het bestand heeft de naam “notes.txt”

StreamReader

- 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.

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!!!

```
sw.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!!!

```
sw.Close();
```

FileStream

- FileStream: meer specifiek aangeven wat de operatie is die mag uitgevoerd worden op het bestand
- FileMode
  - Open: bestaand bestand lezen/schrijven
  - Create: bestand lezen/schrijven, **bestaand bestand wordt overschreven**
  - OpenNew: bestand lezen/schrijven, **bestaand bestand geeft een exception!**
  - Append: toevoegen aan bestand en eventueel creatie indien onbestaand
- FileAccess
  - Read: Enkel lezen
  - Write: Enkel schrijven
  - ReadWrite: Lezen en schrijven

FileStream: meer opties bij het schrijven

- Voeg de juiste namespace toe

```
using System.IO;
```

- Maak FileStream en StreamWriter aan (automatisch in \bin\Debug)

```
FileStream fsw = new FileStream("MyTest.txt", FileMode.Append, FileAccess.Write);
StreamWriter sw = new StreamWriter(fsw); // geef FileStream object door
```

- Naar tekstbestand schrijven

```
sw.WriteLine("Volgorde van getallen:");
for (int i = 0; i < 10; i++)
  sw.Write($"{i} ");
sw.WriteLine(); // lege regel
```

- Bestand sluiten (omgekeerde volgorde als aanmaken)

```
sw.Close();
fsw.Close();
```

FileStream: meer opties bij het lezen

- Voeg de juiste namespace toe

```
using System.IO;
```

- Maak FileStream en StreamReader aan (automatisch in \bin\Debug)

```
FileStream fsr = new FileStream("MyTest.txt", FileMode.Open, FileAccess.Read);
StreamReader sr = new StreamReader(fsr); // geef FileStream object door
```

- Lees het bestand in 1 keer

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

- Bestand sluiten (omgekeerde volgorde als aanmaken)

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

File class

- Static class met hulpmethods voor File I/O
- Enkel gebruiken bij 1 operatie op bestand!
  - Doet per method veiligheidscontroles.
  - Te vaak = traag!

File class

File.Append(pad)	opent en voegt toe aan bestaand bestand
File.Create(pad)	creëert en overschrijft het bestand in opgegeven pad
File.CreateText(pad)	creëert en overschrijft een tekstbestand (UTF-8 encoded)
File.Delete(pad)	verwijdert bestand
File.Copy(pad)	kopieert een bestand
File.Exists(pad)	test of bestand bestaat (true/false)
File.OpenText(pad)	opent een bestaand tekstbestand (UTF-8 encoded)
File.OpenWrite(pad)	opent een bestaand tekstbestand om weg te schrijven
File.OpenWrite(pad)	opent een bestaand tekstbestand om weg te schrijven
File.ReadAllLines(pad)	opent, lees alle lijnen in array en sluit
File.ReadAllLines(pad)	Length geeft aantal regels in bestand
File.ReadAllText(pad)	leest het volledige bestand
File.WriteAllText(pad)	schrijft het volledige bestand weg
File.AppendAllText(pad, string)	voegt tekst aan bestand toe

File class

```
string pad = @"\bestanden\tekst.txt";
if (!File.Exists(pad)) // controleer of bestand nog niet bestaat
{
  //maak StreamWriter en maak bestand aan
  using (StreamWriter sw = File.CreateText(pad))
  {
    sw.WriteLine("Hello");
  }
  // of korter
  File.WriteAllText(pad, "Hello" + Environment.NewLine);
  // Tekst achteraan toevoegen
  File.AppendAllText(pad, "Darkness" + Environment.NewLine);
}
```

- **Using blok**
  - Sluit het bestand automatisch.
  - Zorgt ervoor dat we zelf geen sw.Close() moeten doen.
  - Veiliger, want doet ook sw.Close() bij crashen

File class

- Open het bestand en lees alle tekst

```
string pad = "test.txt";
string alleTekst;
using (StreamReader sr = File.OpenText(pad))
  {
    alleTekst = sr.ReadToEnd();
  }
```

- Kortere manier

```
string pad = "test.txt";
string alleTekst = File.ReadAllText(pad);
```

FileInfo class

- Class waar we hulpobject van maken
- Gebruiken als je meerdere operaties op bestand doet!
  - Doet 1 veiligheidscontrole, enkel bij aanmaken van object

FileInfo class

fi.Exists	Testen of een bestand bestaat
fi.AppendText()	Maakt een StreamWriter aan en voegt tekst toe
fi.CreateText()	Maakt een StreamWriter aan en schrijft tekst weg
fi.OpenText()	Maakt een StreamReader aan en leest tekst
fi.OpenRead()	Maakt een readonly bestand
fi.OpenWrite	Maakt een write-only bestand
fi.Delete()	Verwijdert het bestand.

FileInfo object gebruiken bij het schrijven

```
string pad = "tekst.txt";
FileInfo fi = new FileInfo(pad);
if (!fi.Exists) // controleer of bestand nog niet bestaat
{
  // maak StreamWriter en maak bestand aan
  using (StreamWriter sw = fi.CreateText())
  {
    // schrijf tekst naar bestand
    sw.WriteLine("Hello");
    sw.WriteLine("And");
    sw.WriteLine("Welcome");
  }
}
```

- **Verwijder bestand**
  - fi.Delete();

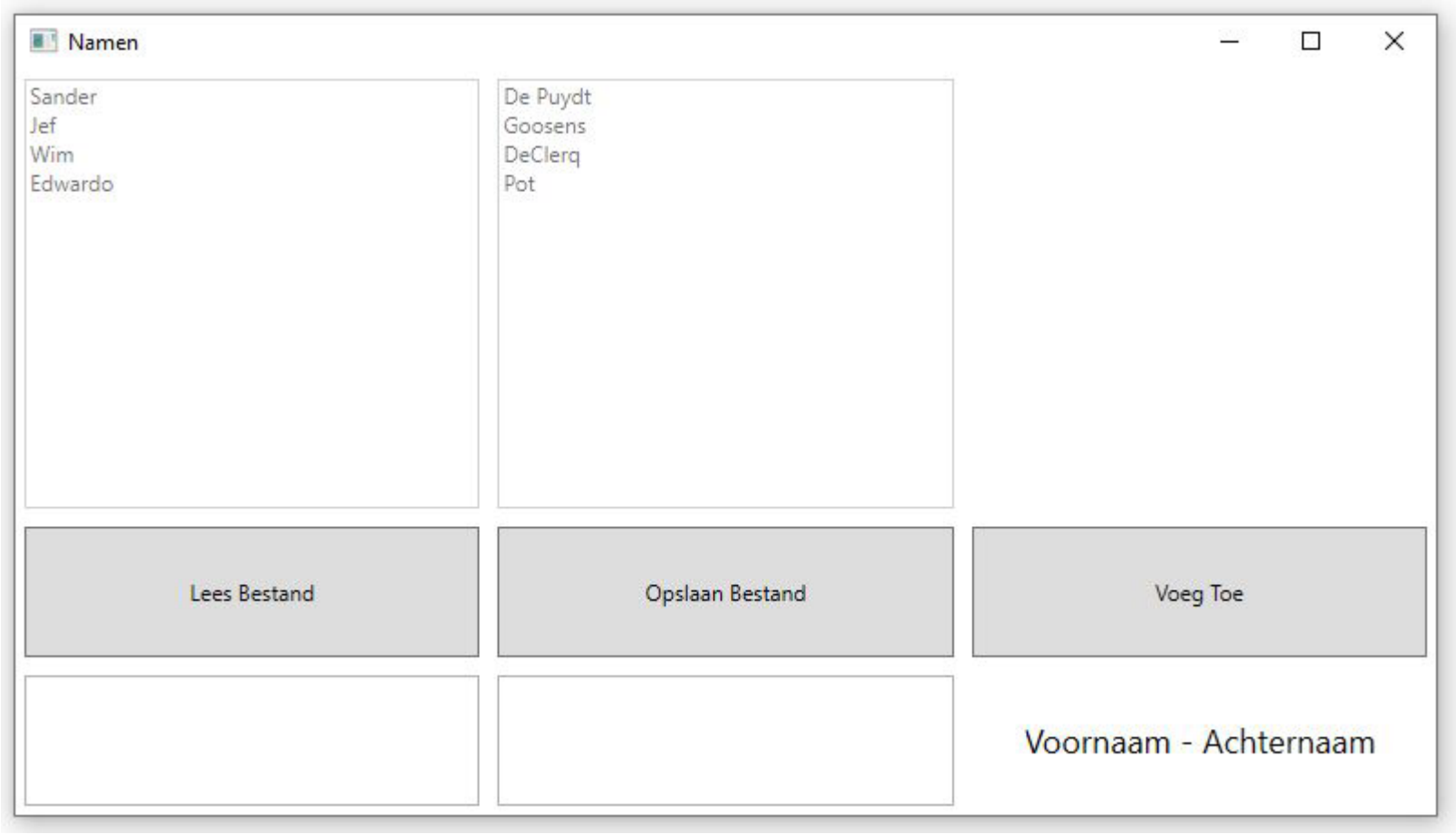
FileInfo object gebruiken bij het lezen

```
string pad = "tekst.txt";
FileInfo fi = new FileInfo(pad);
if (fi.Exists) // enkel bestand lezen als het bestaat
{
  using (StreamReader sr = fi.OpenText()) // maak StreamReader en open bestand
  {
    while (!sr.EndOfStream)
    {
      string lijn = sr.ReadLine();
      Console.WriteLine(lijn);
    }
  }
}
```

Samenvatting

- Stream: een stroom of reeks van bytes
- StreamReader: lezen van bestanden
- StreamWriter: schrijven van bestanden
- FileStream: meer specifiek de operatie beschrijven
- File: statische klasse met hulpmethodes (best 1 bewerking)
- FileInfo: hulpmethodes wanneer je meerdere bewerkingen wilt doen

Oefening “Namen”



Oefening “Namen”

Oefening “Namen”

- Knop “Lees Bestand”

  Leest al de namen uit van het bestand ”personen.txt”

- Knop “Opslaan Bestand”

  Slaat al de namen (die momenteel getoond worden in de twee kolommen) op in het bestand personen.txt

- Knop “Voeg Toe”

  Voegt de voornaam en achternaam toe aan de twee lijsten van namen op het scherm.