

Hand-out

Programmeren

09. Files & IO

1. Inleiding

Gegevens kunnen we opslaan op verschillende manieren. Door gebruik te maken van variabelen kunnen we gegevens opslaan in het geheugen van de computer, echter, als we de computer uit zetten dan zijn we de gegevens kwijt. Door gegevens op te slaan in bestanden kunnen we deze behouden na dat de applicatie is afgesloten of de computer is uitgezet.

Om gegevens op te slaan in bestanden maken we gebruik van de IO namespace uit het .NET framework. Via de File class kunnen we nu in C# gebruik maken van bestanden, hier data in opslaan bewerken en verwijderen.

2. Input Output

IO staat voor Input Output. Input word vaak door de gebruiker verzorgd. Output vaak door het programma. Maar je kunt ook Input hebben van uit een ander medium, bijvoorbeeld een andere website, proces of protocol dat word aangestuurd vanuit een bestand (batch processing) of een database bijvoorbeeld. Maar ook kunnen we verschillende soorten output hebben, iets naar het scherm schrijven is output, maar iets naar een bestand schrijven is ook output.

In deze reader gaan we het strikt en alleen hebben over het opslaan van gegevens in bestanden, en we beginnen heel simpel bij een tekst bestand. Maar we kunnen ook een .ini bestand (config) of een .resx (resource) bestand gebruiken.

Ook een database is in principe niets anders dan een bestand, en ook daar kunnen we natuurlijk in werken, maar meer over databases in een ander hoofdstuk.

3. Bestanden

Gegevens opslaan in een bestand is helemaal niet zo moeilijk, je opent een bestand en als dat bestand nog niet bestaat maak je het aan. Dan open je de stream naar het bestand en schrijf je de data welke je in de stream hebt staan naar het bestand. Dit kan een stuk tekst zijn maar het zou ook de data voor een plaatje kunnen zijn. De drie methodes om tekst naar een bestand te schrijven en te lezen zijn:

- System.IO.File.WriteAllText & ReadAllText
- System.IO.File.WriteAllLines & ReadAllLines
- System.IO.StreamWriter

Nu zijn tekst bestandjes niet bedoelt om plaatjes in op te slaan maar als je het zou doen, en je zou de extensie van het bestand veranderen naar .jpg kan het goed dat je het plaatje kunt openen. Plaatjes zijn dus ook niets meer dan bestanden die een bult code bevatten, deze code is altijd uit te lezen en weg te schrijven. Ook al begrijpen we niet wat er staat.

Om met bestanden te werken dienen we een library in te laden die we niet standaard in het programma hebben zitten. Dit is de IO library en je kunt hem inladen via de volgende namespace.

```
1. using System.IO;
```

Als we vervolgens gebruik willen maken van de functies gebruiken we de File class, als volgt :

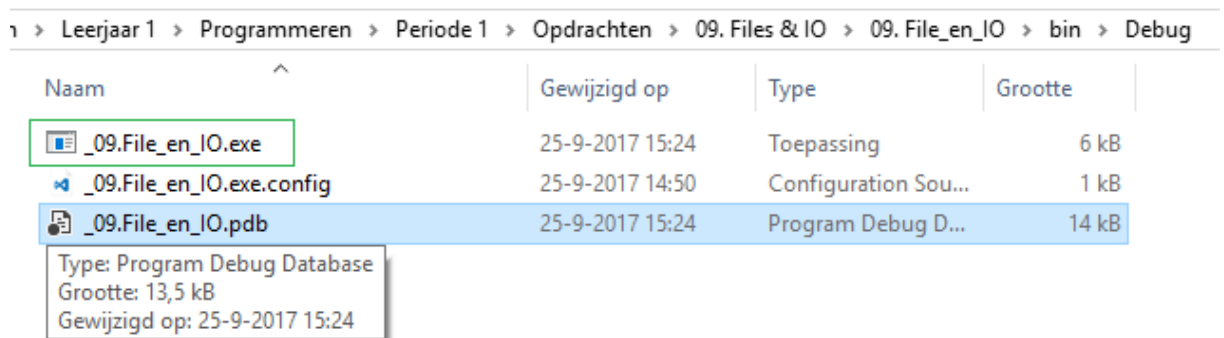
```
1. public void WriteTextToFile(string text)
2. {
3.     string strFileName = "WriteText.txt";
4.     string strPath      = @"C:\Users\Public\";
5.     System.IO.File.WriteAllText(strPath + strFileName, text);
6. }
```




In dit geval zie je het gebruik van een stuk code zonder dat de IO library is ingeladen via het using keyword. Regel 5 laat ons zien dat je een library ook direct aan kunt spreken. Het nadeel hier van is dat we dit constant moeten uitschrijven en dit ons dus meer werk oplevert dan wanneer we de library een maal in de header van het bestand inladen.

4. Bestandslocaties

Om met bestanden te kunnen werken moet je deze wel ergens op kunnen slaan. Dit kan lokaal op je eigen machine, maar je kunt ook een bestand van een andere computer laden via een http of ftp protocol. We gaan in deze reader uit van bestanden die lokaal op je computer staan.

Deze bestanden kun je op twee manieren benaderen, met een absoluut pad of een relatief pad. Absoluut wil zeggen dat je het hele pad beschrijft inclusief de schijfnaam "C:\Users\Public\". Een relatief pad wil zeggen dat je inzet vanaf de locatie waarvan je programma draait, in het geval van een .net programma draait deze in de bin/debug of bin/release folder van project map.

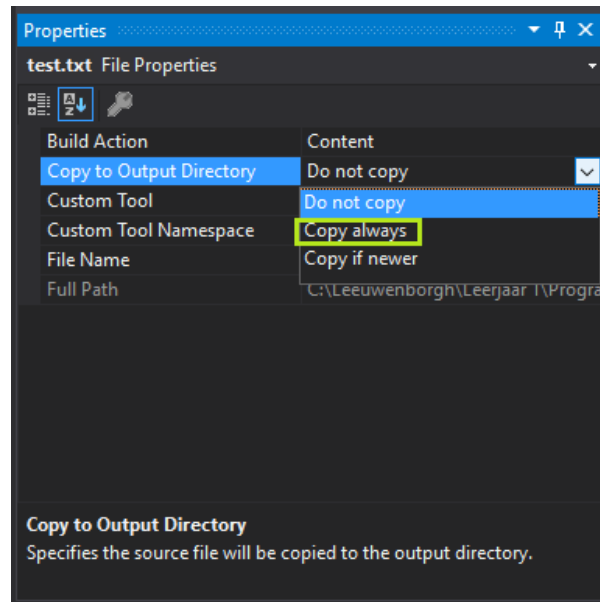


1 > Leerjaar 1 > Programmeren > Periode 1 > Opdrachten > 09. Files & IO > 09. File_en_IO > bin > Debug			
Naam	Gewijzigd op	Type	Grootte
 _09.File_en_IO.exe	25-9-2017 15:24	Toepassing	6 kB
 _09.File_en_IO.exe.config	25-9-2017 14:50	Configuration Sou...	1 kB
 _09.File_en_IO.pdb	25-9-2017 15:24	Program Debug D...	14 kB
Type: Program Debug Database Grootte: 13,5 kB Gewijzigd op: 25-9-2017 15:24			

Een relatief pad ziet er als volgt uit : "../resources/something.txt" en word dus berekend vanuit de locatie waar het programma draait, dit geldt ook voor websites en andere applicaties.

5. Output directory

Wil je nu plaatjes of andere bestanden gebruiken in je applicatie en heb je deze netjes in een mapje gezet binnen je project, let dan op dat je de bestanden naar je output directory mee kopieert anders werken deze niet wanneer je het programma released of op een andere computer zet, meer daar over in een later hoofdstuk.



6. Andere bestanden

In dit hoofdstuk hebben we het alleen over tekstbestanden gehad maar er zijn natuurlijk nog veel meer bestanden en media waar we informatie naar kunnen wegschrijven en van op kunnen halen. Denk hierbij aan CSV (Comma Separated Values) bestanden, Databases en Resources. Meer daar over in Periode 2.

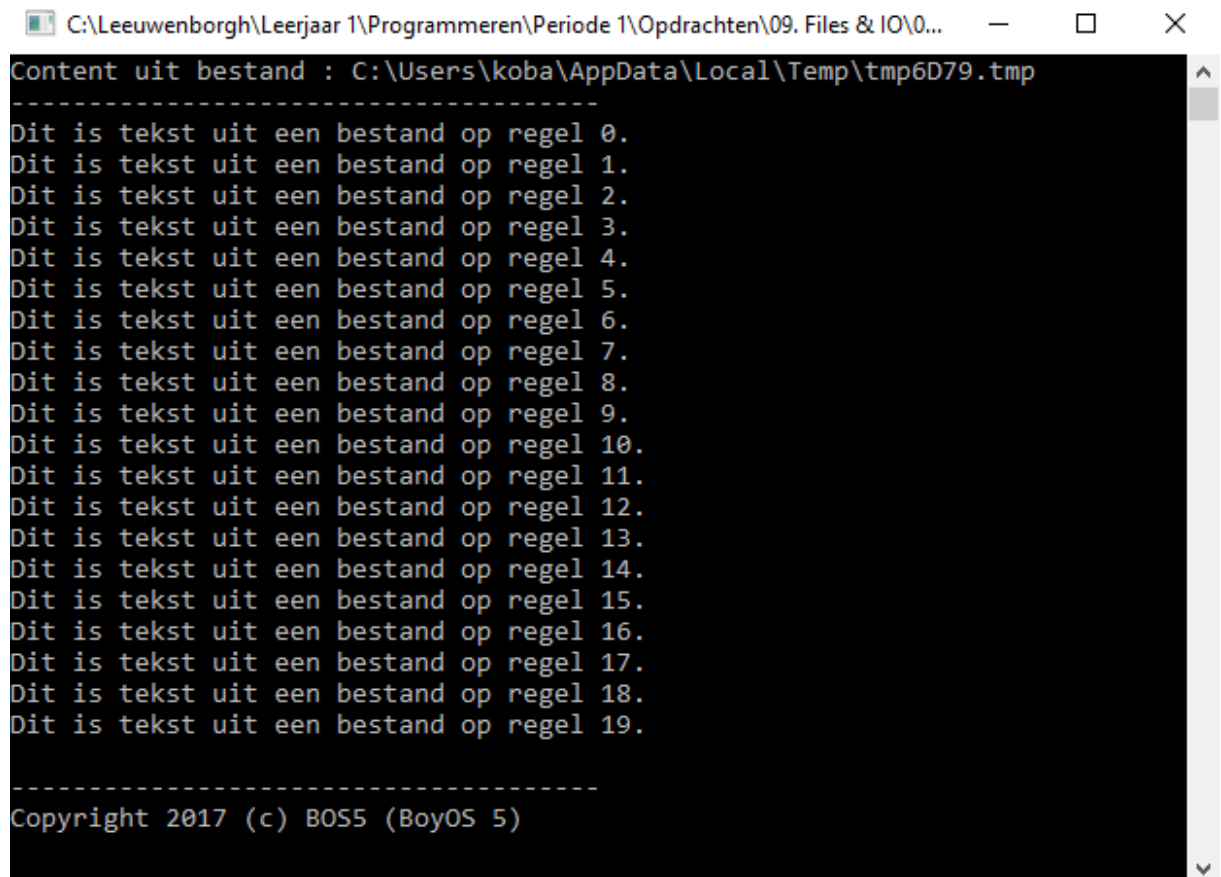
7. Demo

In dit stuk gaan we klassikaal een demo maken. Het is de bedoeling dat je mee doet en zo stapsgewijs houvast krijgt op het onderwerp, maak daar waar nodig is aantekeningen in je schrift of in je kladblok. Als je klaar bent ga je zelf aan de slag dus let goed op!

8. Opdracht

Maak een programma dat een bestand opent, er tekst in schrijft, deze weer uitleest. Je mag zelf weten welke tekst je in het bestand wegschrijft en ook mag je zelf weten welke methode je gebruikt om de tekst naar het bestand te schrijven en weer uit te lezen, het gaat er om dat je gaat begrijpen hoe het werken met bestanden in elkaar steekt.

Als je klaar bent met deze opdracht ga je je werk samen met de docent bekijken. veel succes!



The image shows a Windows File Explorer window with the address bar displaying 'C:\Leeuwenborgh\Leerjaar 1\Programmeren\Periode 1\Opdrachten\09. Files & IO\0...'. Below the address bar, the content of a file is displayed in a dark-themed terminal window. The text in the terminal is as follows:

```
Content uit bestand : C:\Users\koba\AppData\Local\Temp\tmp6D79.tmp
-----
Dit is tekst uit een bestand op regel 0.
Dit is tekst uit een bestand op regel 1.
Dit is tekst uit een bestand op regel 2.
Dit is tekst uit een bestand op regel 3.
Dit is tekst uit een bestand op regel 4.
Dit is tekst uit een bestand op regel 5.
Dit is tekst uit een bestand op regel 6.
Dit is tekst uit een bestand op regel 7.
Dit is tekst uit een bestand op regel 8.
Dit is tekst uit een bestand op regel 9.
Dit is tekst uit een bestand op regel 10.
Dit is tekst uit een bestand op regel 11.
Dit is tekst uit een bestand op regel 12.
Dit is tekst uit een bestand op regel 13.
Dit is tekst uit een bestand op regel 14.
Dit is tekst uit een bestand op regel 15.
Dit is tekst uit een bestand op regel 16.
Dit is tekst uit een bestand op regel 17.
Dit is tekst uit een bestand op regel 18.
Dit is tekst uit een bestand op regel 19.
-----
Copyright 2017 (c) BOS5 (BoyOS 5)
```

De code kun je de volgende les vinden op GitHub (rep 09.-Files & IO)