# Office documenten manipuleren vanuit .NET code met behulp van Open XML

Contents

[Office documenten manipuleren vanuit .NET code met behulp van Open XML 1](#_Toc437972307)

[Duiding 1](#_Toc437972308)

[Open XML en alternatieven 2](#_Toc437972309)

[Open XML 2](#_Toc437972310)

[Office interop API 2](#_Toc437972311)

[VSTO: Visual Studio Tools for Office 2](#_Toc437972312)

[Commerciële pakketten 3](#_Toc437972313)

[Open XML 4](#_Toc437972314)

[Open XML formaat: what’s in it? 4](#_Toc437972315)

[Open XML: tools! 5](#_Toc437972316)

[Open XML: project voorbereiding 6](#_Toc437972317)

[Open XML programmeren: opzoeken van de Excel Worksheet namen 6](#_Toc437972318)

[Open XML programmeren: opzoeken van een Cell waarde 7](#_Toc437972319)

[Open XML programmeren: verwijderen van tekst uit een cell 7](#_Toc437972320)

[Open XML programmeren: schrijf tekst in een Cel 8](#_Toc437972321)

[OpenXMLExcel klasse 8](#_Toc437972322)

[Omtrent snippets en voorbeeldcode 8](#_Toc437972323)

[OpenXMLExcel klasse 8](#_Toc437972324)

[OpenXMLWord klasse 9](#_Toc437972325)

[OpenXML werkwijze 9](#_Toc437972326)

## Duiding

In een administratieve omgeving wordt veel gewerkt met documenten: facturen, diploma’s, telefoonlijsten, examen programma’s, routebeschrijvingen, .. zijn hiervan voorbeelden. In dergelijke documenten is veel informatie terug te vinden.

Een programm kan deze documenten genereren dooe (database-) informatie in een template document in te vullen. In sommige gevallen (vooral Excel) kunnen de documenten uitgelezen worden door een programma om informatie te verzamelen. In deze les zullen we ons beperken tot het manipuleren van Word- en Excel documenten door middel van Open XML.

## Open XML en alternatieven

We zullen in deze les de documenten manipuleren door middel van [Open XML](https://en.wikipedia.org/wiki/Office_Open_XML). Dit is zeker niet de enige mogelijke manier. Een aantal alternatieven op een rijtje:

### Open XML

De recentere office versies bewaren hun documenten in een zip/XML formaat. Deze kunnen gemanipuleerd worden zonder dat office geïnstalleerd werd op het uitvoerende toestel. Ook de snelheid zou volgens de documentatie beduidend hoger zijn. Wanneer we de OpenXML SDK gebruiken is de beprogrammering van deze XML files doenbaar (maar moeilijker dan de office interop benadering). Voordelen van deze benadering zijn:

* Sneller dan de office interop api benadering
* Geen office installatie nodig

Nadelen van deze benadering zijn:

* Deze manier van werken is enkel bruikbaar in een klassieke desktop omgeving (niet in UWP)
* Het vergt enige moeite om zich in te werken in de mogelijkheden van OpenXML

### Office interop API

Er bestaat een .NET wrapper rond de office COM API waarmee de office omgeving als programmeer- model (een set klassess met methodes) wordt binnengehaald. Deze manier van werken heeft drie nadelen:

* Office moet geïnstalleerd zijn op de uitvoerende computer
* Het uitvoeren van dergelijk programma start office en voert de bewerking ongeveer uit zoals een gebruiker dit zou doen. Deze manier van werken is traag;
* Deze manier van werken is enkel bruikbaar in een klassieke desktop omgeving (niet in UWP)

Het grote voordeel van deze manier van werken:

* Het programmeert gemakkelijk, en doordat office (VBA) macro’s kan opnemen is het mogelijk, zonder het object model volledig te doorgronden, snel een werkend programma te ontwikkelen
  + Het programmeren van office interop gebeurt best via VB.NET. Deze taal is een variant van de macro taal VBA (waardoor opgenomen macro’s gemakkelijk omgezet kunnen worden) en de late binding is een plus wanneer we de programma’s office versie onafhankelijk maken.

### VSTO: Visual Studio Tools for Office

Dit is een .NET programmer omgeving die de office pakketten als UI gebruikt. U kan zelf controls aan de Office ribbons toevoegen en logica koppelen aan control events. Ik vond dit altijd een technologie met veel potentieel, in de praktijk zag ik er weinig voorbeelden van.

Voordelen:

* De eindgebruiker opent een office document (en niet een nieuw programma) waarbinnen de programma logica het office document aanstuurt

Nadelen

* Het lijkt nooit aan te slaan. Eén van de potentële nadelen is het moeilijk inschatbaar zijn van de inspanning om van één office versie naar een andere over te stappen.
* Voor zover ik weet is deze oplossing niet bruikbaar in een UWP context

### Commerciële pakketten

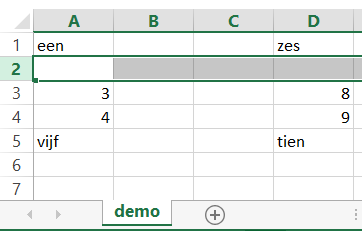
Hiervan zijn er ettelijke die snel en degelijk werken (hear say).

Volgens de documentatie zijn sommigen ook bruikbaar in een UWP omgeving

## Open XML

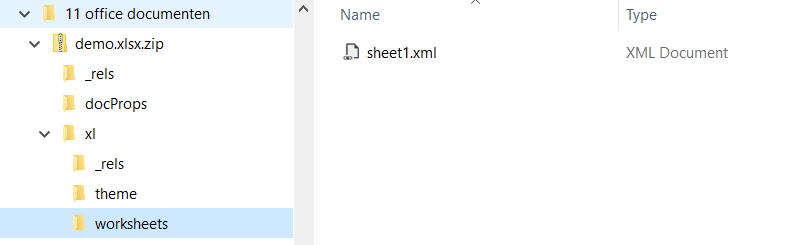
Mijn voornaamste bron van informatie voor deze topic is de PluralSight cursus [Introduction to Microsoft Office Open XML](http://app.pluralsight.com/training/player?course=introduction-microsoft-office-open-xml&author=ken-getz&name=introduction-microsoft-office-open-xml-m1-intro&clip=3&mode=live) geweest (tot nu toe werkte ik zelf vooral met de office interop API). De introductie in deze tekst is een verkorte versie van de plural sight cursus. Een flink aantal ‘how to’ voorbeelden kan u terugvinden op [How do I ..? (Open XML SDK)](https://msdn.microsoft.com/EN-US/library/office/bb491088.aspx).

### Open XML formaat: what’s in it?

Open XML verpakt de office files als ZIP containers waarbinnen, door middel van XML- files de inhoud van de document wordt beschreven (De xml- inhoud verklaart de extra x in de suffix van de office documenten).

Veronderstel volgende Excel file:

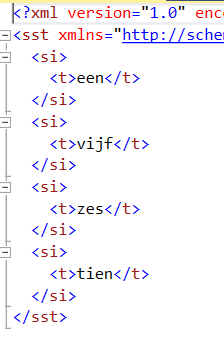
* Er zijn tekst waardes: een, vijf, ..
* Er zijn numerieke waardes: 3, 4, ..
* Er is één worksheet met de naam demo
* Deze Excel- file wordt bewaard als demo.xlsx

Indien we de filenaam wijzigen tot demo.xlsx.zip is deze file via de verkenner ondervraagbaar:

* De xl folder bevat een worksheets folder waarbinnen we sheet1.xml terugvinden.

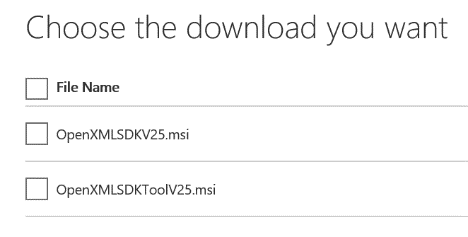
Indien we de sheet1.xml file openen (Visual Studio is mijn favoriete XML editor in deze omstandigheden) zien we rechtse structuur:

* Een worksheet heeft (onder andere) een sheetData tag
* Een sheetData tag bevat verschillende row tags
  + Enkel de rijen waarvoor er ingevulde cellen zijn komen in de sheetData tag voor
* Een row tag bevat verschillende c (Cell) tags
  + Enkel de cellen die werden ingevuld komen als tag voor in een row
* Een cel heeft een type (het t- attribuut). Indien de cel een tekst bevat vinden we toch een numerieke waarde (dit is een verwijzing naar een shared string waarde)
* Het default cel type is een getal. Dit is dus ook het type indien er geen type vermeld wordt.

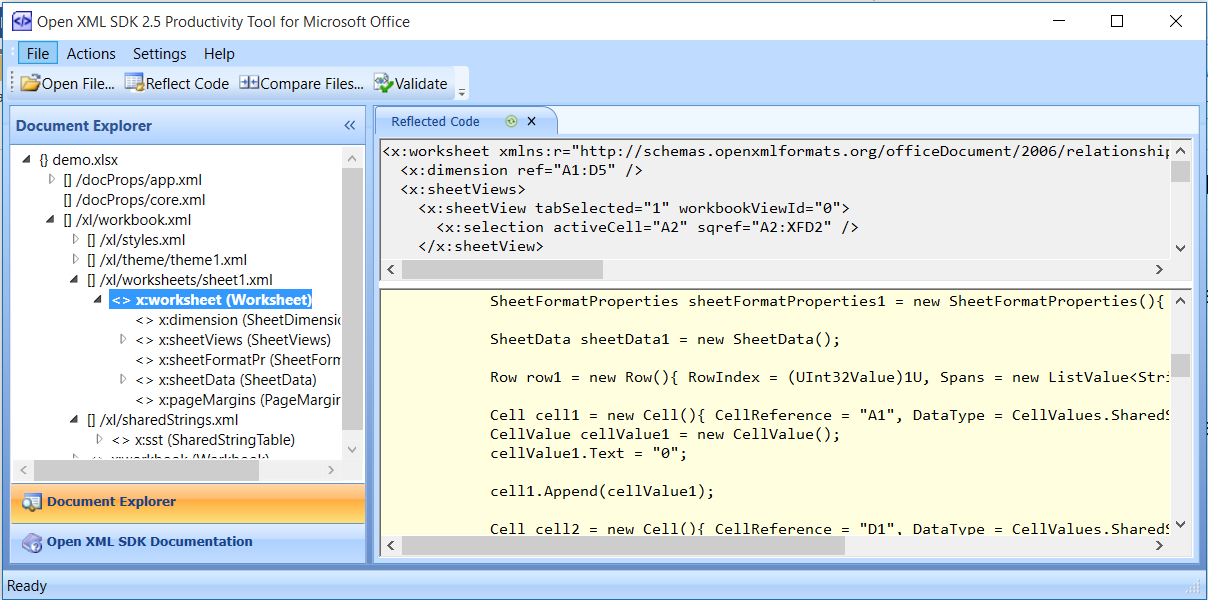
In de gezipte file structuur vinden we ook een SharedStrings.xml file terug:

* Hierin worden de tekstwaardes (literals) bijgehouden. Op deze manier moeten we geen dubbels bijhouden. De getalwaarde in de cell stemt overeen met de index (positie) in de SharedStrings file. Indien we deze informatie koppelen aan de eerste cel informatie in sheetdata (vorige image) dan kunnen we achterhalen dat het woord ‘een’ in de eerste cel wordt getoond:
  + Het type van de cel op adres A1 is s: een shared string
  + De value is 0 (voor dit type betekent dit de index in de SharedStrings file
  + De eerste waarde is een, wat de te tonen tekst is.

### Open XML: tools!

Om Open XML vlot te kunnen gebruiken zullen we de Open XML SDK installeren (huidige versie is 2.5). Even googlen brengt u op een download link met daarin twee onderdelen:

* OpenXMLSDKV25.msi levert de dll’s die we in onze programma’s zullen gebruiken
* OpenXMLSDKToolV25.msi levert een tool waarmee we gemakkelijker een Open XML document kunnen onderzoeken.

Een screenshot van de Open XML SDK 2.5 Productivity Tool:

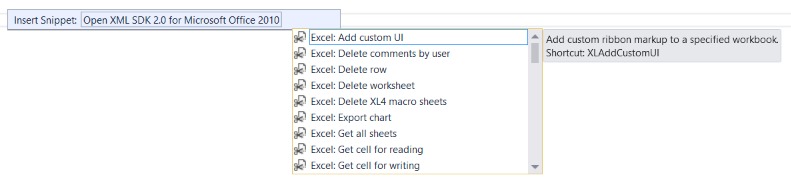
De tool toont volgende informatie:

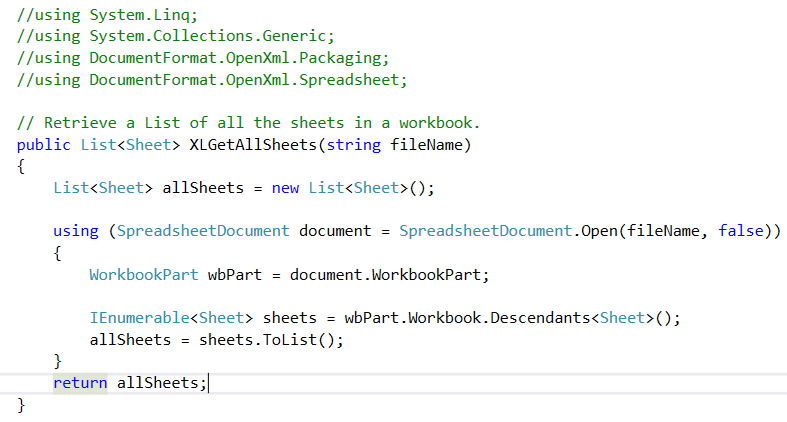
* De folderstructuur
* De XML code
* De overeenkomstige C# code waarmee het gekozen XML- onderdeel kan aangemaakt worden. Deze C# code maakt gebruik van klasses die in de Open XML SDK worden gedefinieerd.
* Download en installeer [de code snippets](https://www.microsoft.com/en-us/download/details.aspx?id=17985) (jammer: SDK 2.0 en VS 2010, maar ze werken nog). Met behulp van de Code Snippets Manager kan u deze snippets in uw Visual Studio omgeving opnemen (let er op dat u de juiste programmeer taal kiest). De toets combinaties ctrl-k ctrl-x zullen u tijdens het programmeren toelaten om een snippet te kiezen.

### Open XML: project voorbereiding

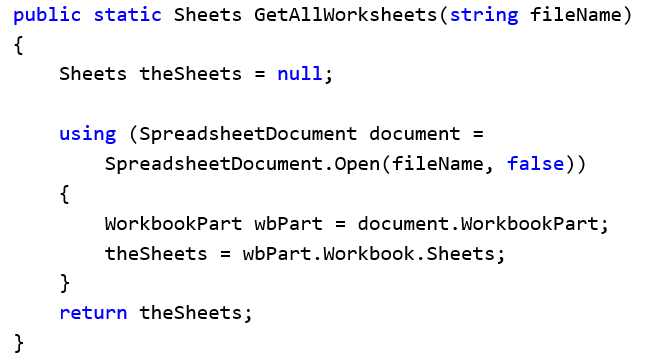
* Installeer de Open XML tools zoals hierboven beschreven (SDK en snippets)
* Maak een nieuw **WPF** project OpenXMLDemo;
* Installeer het NuGet Package DocumentFormat.OpenXML

### Open XML programmeren: opzoeken van de Excel Worksheet namen

* Er bestaat een snippet! 
* Deze snippet genereert werkende (sub optimale) code:



* Indien we [How to: Retrieve a list of the worksheets in a spreadsheet document (Open XML SDK)](https://msdn.microsoft.com/EN-US/library/office/jj618414.aspx) er op nalezen zien we onderstaande code:



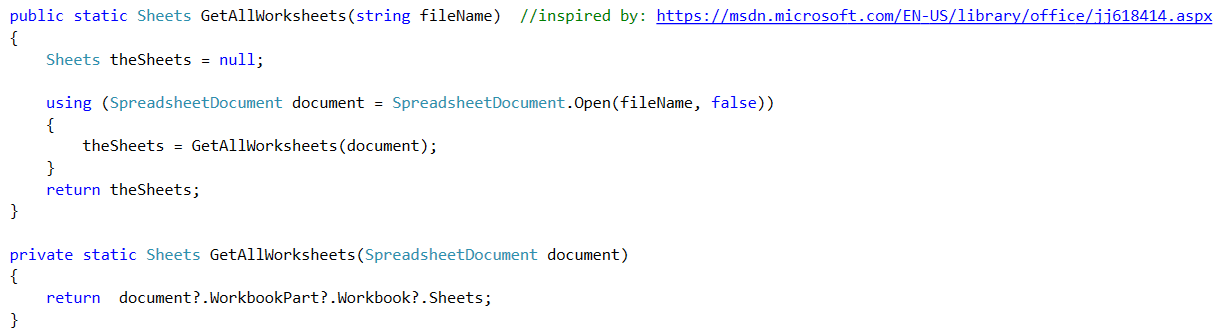
Het verschil tussen beide voorbeeld codes is te verklaren door het versie verschil van de SDK waarvoor elk werd ontwikkeld: de snippets werden ontwikkeld voor SDK 2.0 en de How- to’s zijn geschreven voor SDK 2.5. Jammer genoeg zijn er geen snippets geschreven voor SDK 2.5 ☹, want:

* Snippets zijn gemakkelijker te gebruiken dan het opzoeken en copiëren van code
* Verouderde snippets zijn wellicht een garantie op suboptimale code

Misschien moeten we zelf wat snippets voorzien?

#### Nadeel van de voorbeeldcode

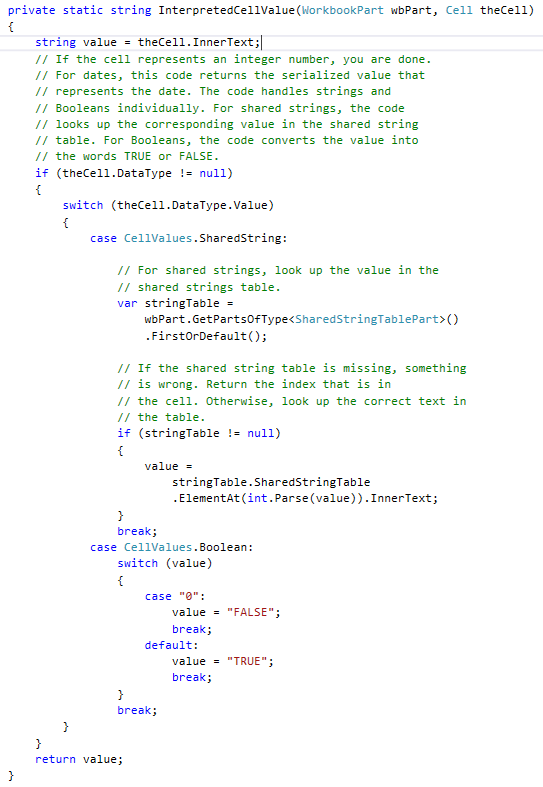
De voorbeeld code die we terugvinden (zowel de snippets als de howto’s) zullen alle nodige logica uitvoeren: we vertrekken van een (full) filename en geven het resultaat terug. In veel gevallen zullen we op dezelfde file opeenvolgende bewerkingen uitvoeren en zullen we de geopende file bijhouden. Mijn voorkeur code ziet er dan ook als volgt uit:



We voorzien twee signaturen van dezelfde functie:

* Er is een signatuur die werkt met een (full) file name
* Deze maakt gebruikt van een signatuur die werkt met een spreadsheet document

### Open XML programmeren: opzoeken van een Cell waarde

Dit onderdeel is gebaseerd op [How to: Retrieve the values of cells in a spreadsheet document (Open XML SDK)](https://msdn.microsoft.com/EN-US/library/office/hh298534.aspx).

Wat we op basis van deze tekst onthouden:

* Indien de tekst in een veld een numerieke waarde is dan is de innertext van de cell de string voorstelling van deze waarde
* Indien de tekst in een veld een tekst waarde is dan moet de effectieve waarde in een lijst worden opgezocht (SharedStrings). De innertext van de cel is dan de index (het volgnummer) in de shared strings lijst.
* Indien de tekst in een veld een Boolean waarde bevat dan is de waarde 0 gelijk aan FALSE, alle andere waardes zijn TRUE.
* Een datum heeft als waarde een getal: [het aantal verstreken dagen sinds 1900](http://www.cpearson.com/excel/datetime.htm).

### Open XML programmeren: verwijderen van tekst uit een cell

Dit onderdeel is gebaseerd op [How to: Delete text from a cell in a spreadsheet document (Open XML SDK)](https://msdn.microsoft.com/EN-US/library/office/cc797480.aspx).

De extra complexiteit die we hier moeten oplossen is het onderhouden van de SharedStrings file: indien geen enkele andere cel deze waarde als tekst heeft, dan moet deze waarde uit de SharedStrings verwijderd worden. Dit betekent op zijn beurt dat de index van alle erna komende waardes met 1 verminderd moet worden in de cell values.

De voorbeeld code die u op de website vind voorziet de code om de SharedStrings te onderhouden, maar roept deze niet op.

### Open XML programmeren: schrijf tekst in een Cel

Dit onderdeel is gebaseerd op [How to: Insert text into a cell in a spreadsheet document (Open XML SDK)](https://msdn.microsoft.com/EN-US/library/office/cc861607.aspx).

Belangrijk bij het schrijven van waardes in een Excel is het feit dat de rijen opeenvolgend in de SheetData moeten opgenomen zijn: indien rij 21 voor rij 20 wordt geplaatst is de Excel corrupt!

De voorbeeld code waarop deze tekst is gebaseerd houdt hiermee geen rekening.

## OpenXMLExcel klasse

### Omtrent snippets en voorbeeldcode

De snippets en de voorbeeldcode zijn goed om testen uit te voeren (werkt de technologie?) maar hebben een aantal beperkingen in een real- life omgeving:

* De verschillende voorbeelden hebben gelijkaardige maar licht verschillende code:
  + Zowel bij het ophalen van een Tekst waarde als het verwijderen van een tekst waarde moet een Cell- object worden opgehaald
* Indien meerdere bewerkingen op dezelfde Excel file of Excel sheet moeten uitgevoerd worden zijn de voorbeeldcodes hiervoor niet geoptimaliseerd:
  + Elk voorbeeld opent opnieuw het Excel document, bepaalt opnieuw de Sheet enz. Indien we meerdere bewerkingen op hetzelfde object wensen uit te voeren kunnen we dit optimaliseren door het document slechts één keer te openen en de sheet slechts één keer te bepalen.
  + Natuurlijk kan het ook interessant zijn om andere object types die reeds werden bepaald niet opnieuw te bepalen

### OpenXMLExcel klasse

Op leho vindt u een klasse OpenXMLExcel met hierin een beperkte set routines om Excel bestanden te manipuleren:

* XLGetCell om een cell op te vragen
* XLGetCellValue om gegevens op te vragen
* XLInsertText om tekst in een cell te plaatsen
* XLDeleteTextFromCell om tekst uit een cell te verwijderen.

U mag uiteraard gebruik maken van deze klasse om basis Excel acties uit te voeren of voorbeeld code te copiëren. In een productieomgeving veronderstel ik dat deze eenvoudige OpenXMLExcel klasse verder zou uitgebreid worden.

Deze eenvoudige klasse heeft als voornaamste beperking dat ze enkel tekst waardes kan verwerken. In een meer realistische benadering zullen uiteraard ook numerieke en andere waardes gebruikt worden.

## OpenXMLWord klasse

Deze klasse werd gelijkaardig aan de OpenXMLExcel klasse opgezet. Er is slechts één methode voorzien, die gebruikt kan worden om bookmark tekst in een Word document te plaatsen.

## OpenXML werkwijze

Het is niet de bedoeling (maar wel mogelijk) om een document van scratch op te bouwen. We gaan als volgt te werk:

* Maak een office document (Excel of Word) waarin de nodige *kadertekst* aanwezig is.
* Indien het een Word document betreft plaatsen we bookmarks waar er door het programma tekst moet worden toegevoegd.
* Verzorg de opmaak van het document, ook van de in te vullen waardes (het is veel gemakkelijker om deze opmaak in de oorspronkelijke versie te voorzien dan deze via het programma toe te voegen).
* Copieer dit bron document naar een copy file waarbinnen de waardes zullen toegevoegd worden.
  + Eerst copiëren we de file, achteraf passen we de copy aan. Op deze manier blijft het bronbestand altijd correct, ook indien het programma halfweg crasht.