Version 2.0.2.0  
egonl  
February 2019

SharpDocx

Summary

Generating documents with SharpDocx is a two-step process. First you create a view in Word. A view is a Word document which also contains C# code. Code can be inserted anywhere, e.g. 15/03/2020 09:26:16 would insert the current date and time.

The next step is to create documents based on this view. This requires two lines of code:

var document = DocumentFactory.Create("view.cs.docx");

document.Generate("output.docx");

Out of the box SharpDocx supports inserting text, tables, images and more. This tutorial shows you how.

If you want, you can specify a view model to be used in your view. Then you could write things like < % foreach (var item in Model.MyList) { % >. See the Model sample.

If you want to do something that's not supported by SharpDocx, you can do so by creating your own document subclass. See the Inheritance example.

SharpDocx is inspired by Web technologies like ASP.NET and JSP. Developers familiar with those technologies should feel right at home.

Generating documents with SharpDocx can be very fast: a slightly modified Model sample produced 25 documents per second on my modest laptop. That’s 1500 documents per minute. Single threaded.

Contents

[The basics 3](#_Toc32010234)

[The Write method 3](#_Toc32010235)

[Conditional content 4](#_Toc32010236)

[Loops 5](#_Toc32010237)

[Nested loops 6](#_Toc32010238)

[Loops, tables and the AppendRow method 7](#_Toc32010239)

[Combining loops, text blocks and tables 8](#_Toc32010240)

[Images 9](#_Toc32010241)

[Replacing text 10](#_Toc32010242)

[Referencing assemblies and importing namespaces 11](#_Toc32010243)

[The Map 12](#_Toc32010244)

[The SharpDocx solution 13](#_Toc32010245)

# The basics

At any point in the text you can insert C# statements. Like right here.

The result looks like this:



## The Write method

If you want to display the value of i, you can use the Write method. Right now, i is 1.

This will show:



There’s also a shorthand notation for the Write method: i is still 1.

This results in:



You can insert line breaks by using ‘\n’:

This paragraph  
contains two  
line breaks.

# Conditional content

You can use an if statement to display conditional content.

This will be displayed.

In this case, any formatting will be lost because the code parser ignores any formatting.

If you want to conditionally display text with formatting, use a *text block*: text between two code blocks and placed between curly brackets:

This **will** *also* be displayed, but *with* formatting.

If you want, you can span multiple elements in a text block. E.g.

The diverging pronunciation of tomato (though not so much potato) is primarily one of regional dialect.

The pronunciation 'tuh-MAH-toh' is the standard pronunciation in the UK and is accepted in the US regions of New England along with parts of the lower East Coast, while 'tuh-MAY-toh' is found almost everywhere else.

Else statements are at the moment not supported by SharpDocx. Instead, use another if statement.

# Loops

You can add repeating text blocks to a document like this:

C:\Users\la_113288\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Infobox_info_icon.emfThe value of i is **1**.i *squared* is **1**

C:\Users\la_113288\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Infobox_info_icon.emfThe value of i is **2**.i *squared* is **4**

C:\Users\la_113288\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Infobox_info_icon.emfThe value of i is **3**.i *squared* is **9**

C:\Users\la_113288\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Infobox_info_icon.emfThe value of i is **4**.i *squared* is **16**

C:\Users\la_113288\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Infobox_info_icon.emfThe value of i is **5**.i *squared* is **25**

C:\Users\la_113288\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Infobox_info_icon.emfThe value of i is **6**.i *squared* is **36**

C:\Users\la_113288\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Infobox_info_icon.emfThe value of i is **7**.i *squared* is **49**

C:\Users\la_113288\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Infobox_info_icon.emfThe value of i is **8**.i *squared* is **64**

C:\Users\la_113288\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Infobox_info_icon.emfThe value of i is **9**.i *squared* is **81**

C:\Users\la_113288\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Infobox_info_icon.emfThe value of i is **10**.i *squared* is **100**

C:\Users\la_113288\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Infobox_info_icon.emfThe value of i is **11**.i *squared* is **121**

C:\Users\la_113288\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Infobox_info_icon.emfThe value of i is **12**.i *squared* is **144**

Nested loops  
Loops can also be nested.

### Multiples of 1

1 \* 1 = 1

1 \* 2 = 2

1 \* 3 = 3

**Note:** 3 is divisible by 3.

### Multiples of 2

2 \* 1 = 2

2 \* 2 = 4

2 \* 3 = 6

**Note:** 6 is divisible by 3.

### Multiples of 3

3 \* 1 = 3

**Note:** 3 is divisible by 3.

3 \* 2 = 6

**Note:** 6 is divisible by 3.

3 \* 3 = 9

**Note:** 9 is divisible by 3.

### Multiples of 4

4 \* 1 = 4

4 \* 2 = 8

4 \* 3 = 12

**Note:** 12 is divisible by 3.

# Loops, tables and the AppendRow method

Sometimes you do want a loop, but you don’t want a repeating text block. For example, you just want to append rows to a table, but you don’t want to repeat the table itself. In this case, use {! instead of {.

This text and table do NOT repeat, because we used {!. However, a couple of rows do get appended to the table by using the AppendRow method.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i \* 1 | i \* 2 | i \* 3 | i \* 4 | i \* 5 | i \* 6 | i \* 7 | i \* 8 | i \* 9 | i \* 10 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 |
| 3 | 6 | 9 | 12 | 15 | 18 | 21 | 24 | 27 | 30 |
| 4 | 8 | 12 | 16 | 20 | 24 | 28 | 32 | 36 | 40 |
| 5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 45 | 50 |
| 6 | 12 | 18 | 24 | 30 | 36 | 42 | 48 | 54 | 60 |
| 7 | 14 | 21 | 28 | 35 | 42 | 49 | 56 | 63 | 70 |
| 8 | 16 | 24 | 32 | 40 | 48 | 56 | 64 | 72 | 80 |
| 9 | 18 | 27 | 36 | 45 | 54 | 63 | 72 | 81 | 90 |
| 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 |
| 11 | 22 | 33 | 44 | 55 | 66 | 77 | 88 | 99 | 110 |
| 12 | 24 | 36 | 48 | 60 | 72 | 84 | 96 | 108 | 120 |
| 13 | 26 | 39 | 52 | 65 | 78 | 91 | 104 | 117 | 130 |
| 14 | 28 | 42 | 56 | 70 | 84 | 98 | 112 | 126 | 140 |
| 15 | 30 | 45 | 60 | 75 | 90 | 105 | 120 | 135 | 150 |
| 16 | 32 | 48 | 64 | 80 | 96 | 112 | 128 | 144 | 160 |
| 17 | 34 | 51 | 68 | 85 | 102 | 119 | 136 | 153 | 170 |
| 18 | 36 | 54 | 72 | 90 | 108 | 126 | 144 | 162 | 180 |
| 19 | 38 | 57 | 76 | 95 | 114 | 133 | 152 | 171 | 190 |
| 20 | 40 | 60 | 80 | 100 | 120 | 140 | 160 | 180 | 200 |
| 21 | 42 | 63 | 84 | 105 | 126 | 147 | 168 | 189 | 210 |
| 22 | 44 | 66 | 88 | 110 | 132 | 154 | 176 | 198 | 220 |
| 23 | 46 | 69 | 92 | 115 | 138 | 161 | 184 | 207 | 230 |
| 24 | 48 | 72 | 96 | 120 | 144 | 168 | 192 | 216 | 240 |
| 25 | 50 | 75 | 100 | 125 | 150 | 175 | 200 | 225 | 250 |
| 26 | 52 | 78 | 104 | 130 | 156 | 182 | 208 | 234 | 260 |
| 27 | 54 | 81 | 108 | 135 | 162 | 189 | 216 | 243 | 270 |
| 28 | 56 | 84 | 112 | 140 | 168 | 196 | 224 | 252 | 280 |
| 29 | 58 | 87 | 116 | 145 | 174 | 203 | 232 | 261 | 290 |
| 30 | 60 | 90 | 120 | 150 | 180 | 210 | 240 | 270 | 300 |

## Combining loops, text blocks and tables

You can nest tables in text blocks in order to create multiple tables. Note that the inner loop does not create a repeating text block, but does append rows.

### Multiples of 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| i |  | j |  | i \* j |
| 1 | \* | 1 | = | 1 |
| 1 | \* | 2 | = | 2 |
| 1 | \* | 3 | = | 3\* |
| 1 | \* | 4 | = | 4 |
| 1 | \* | 5 | = | 5 |
| 1 | \* | 6 | = | 6\* |
|  |  |  |  | \* Divisible by 3 |

### Multiples of 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| i |  | j |  | i \* j |
| 2 | \* | 1 | = | 2 |
| 2 | \* | 2 | = | 4 |
| 2 | \* | 3 | = | 6\* |
| 2 | \* | 4 | = | 8 |
| 2 | \* | 5 | = | 10 |
| 2 | \* | 6 | = | 12\* |
|  |  |  |  | \* Divisible by 3 |

### Multiples of 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| i |  | j |  | i \* j |
| 3 | \* | 1 | = | 3\* |
| 3 | \* | 2 | = | 6\* |
| 3 | \* | 3 | = | 9\* |
| 3 | \* | 4 | = | 12\* |
| 3 | \* | 5 | = | 15\* |
| 3 | \* | 6 | = | 18\* |
|  |  |  |  | \* Divisible by 3 |

# Images

Insert images using the Image method.

​

If only a file name is specified, SharpDocx searches this file in a directory specified by the ImageDirectory property. Right now this property has been set to ‘E:\Projekty\Stoorm5\SharpDocx\Samples\SampleProjects\Tutorial\bin\Debug\net45/../../../../../Images’.

The Image method accepts a second optional parameter that specifies the relative size of the image. Here’s  at 15%.

Images that are too wide to be displayed at 100% are automatically scaled back. Here’s an example:



# Replacing text

If you want to replace text, you can use the Replace method.

This will replace *all* occurrences of the specified string.[[1]](#footnote-2)

Here’s the **replaced text**. And here’s some more replaced text.

# Referencing assemblies and importing namespaces

If you want to use specific types in a view, you can use the Assembly and Import directives to get access to them. Directives look like regular code blocks, but they always start with < %@.

Reference an assembly with the Assembly directive.

Import namespaces with the Import directive.

In C# you would write:

using System.Xml.Linq;

Now we can use types in System.Xml.Linq. Let’s read some news.

**Oracle Criticized For Questioning Google's Supporters In Java API Copyright Case**America's Supreme Court will soon decide whether Google infringed on a copyright that Oracle says it holds on the APIs of Java. But this week Oracle's executive vice president also wrote a blog post a…

**MIT Celebrates 'Pi Day' With Star Wars-Themed Video**DevNull127 writes: MIT has a long-standing tradition. High school seniors who've been accepted into next year's class at MIT are informed on Pi Day (March 14th &mdash; that is, 3/14). And each year th…

**Study Finds High Demand for Go and AR/VR Programmers, While Python Remains Favorite Language**The tech jobs marketplace at Hired.com crunched their data on more than 400,000 interview requests and job offers over the last year to produce their annual "State of Software Engineers" report. Among…

**US Regulators Quickly Approve Roche's New and Faster COVID-19 Test**schwit1 quotes the International Business Times:  
Swiss pharmaceutical giant Roche announced Friday it had received emergency approval from U.S. regulators for a new and much faster test for diagnosing…

**How One Man Lost His Life Savings in a SIM Hack**Long-time Slashdot reader smooth wombat quotes CNN: Robert Ross was sitting in his San Francisco home office in October 2018 when he noticed the bars on his phone had disappeared and he had no cell co…

**Responding to School Closings, Microsoft Offers Free 'Minecraft Education Edition'**Long-time Slashdot reader theodp spotted this announcement at Minecraft.net:  
As COVID-19 concerns prompt school closures around the world, many educators are turning to remote learning to keep student…

**Also Cancelled Over Coronavirus: FIRST Robotics Competition and Stanford's Final Exams**Stanford University announced strict new rules for undergraduates after a student tested positive for COVID-19:  
"The university already went to online classes last week," notes the Los Angeles Times. …

**Group Builds Massive New Library of Censored Articles Inside Minecraft**In countries where websites, blogs and a free press are strictly limited, Minecraft "is still accessible by everyone," notes the official official web site for Reporters Without Borders (an internatio…

**Coronavirus Causes a Bicycling Boom in New York City**An anonymous reader quotes the nonprofit environmental magazine Grist:  
  
On Sunday, New York City Mayor Bill de Blasio unveiled a new set of guidelines for citizens hoping to help contain the burgeonin…

**How Intel is Changing the Future of Power Supplies With Its ATX12VO Spec**"We don't often talk about power supplies, but Intel's new ATX12VO spec &mdash; that's an 'O' for 'Oscar,' not a zero &mdash; will start appearing soon in pre-built PCs from OEMs and system integrator…

**900 Million Secrets From 8 Years of 'Whisper' App Were Left Exposed Online**Long-time Slashdot reader AmiMoJo shares a startling report from the Washington Post:  
Whisper, the secret-sharing app that called itself the "safest place on the Internet," left years of users' most i…

**Should Google Notify Web Sites About Right-to-Be-Forgotten Requests?**An anonymous reader quotes VentureBeat:  
Sweden's Data Protection Authority (DPA) has slapped Google with a 75 million kronor ($8 million) fine for "failure to comply" with Europe's General Data Protec…

**Researchers Create Focus-Free Camera With New Flat Lens**Using a single lens that is about one-thousandth of an inch thick, researchers have created a camera that does not require focusing. Phys.Org reports: "Our flat lenses can drastically reduce the weigh…

**How Koenigsegg's 2.0-Liter No-Camshaft Engine Makes 600 Horsepower**ArchieBunker quotes a report from Road &amp; Track: You can always count on Koenigsegg to do things differently. Take the Swedish brand's newest car, the Gemera, a 1700-hp four-seat hybrid grand toure…

**South Sudan Is Building Its Electric Grid Virtually From Scratch**After years of civil war, South Sudan is beginning to build the nation's electric sector from the ground up. "With only a handful of oil-fired power plants and crumbling poles and wires in place, the …

In a real world scenario you wouldn’t fetch data or have this much code in a view. But hey, this is just an example.

# The Map

The Map maps OpenXmlElements to plain text and vice versa. It’s being used internally by the Replace method and for finding the C# code in views, among other things. At the moment Map.Text looks something like this:

Version 2.0.2.0  
egonl  
February 2019  
Version 2.0.2.0  
egonl  
February 2019  
SharpDocx  
SharpDocx  
  
  
Summary  
Generating documents with SharpDocx is a two-step process. First you create a view in Word. A view is a Word document which also contains C# code. Code can be inserted anywhere, e.g. 15/03/2020 09:26:16 would insert the current date and time.  
The next step is to create documents based on this view. This requires two lines of code:  
 var document = DocumentFactory.Create("view.cs.docx");  
 do …

The Map might be handy when you want to search the document for text.

# The SharpDocx solution

### Building the example programs

The Tutorial, Inheritance and Model samples will by default be build for .NET Framework 3.5, 4.5 and .NET Core 2.0. The .NET Core 2.0 builds will use the .NET Standard 2.0 version of SharpDocx.

The Tutorial and Inheritance samples will by default run in .NET Framework 4.5. The Model sample will by default run in .NET Core 2.0. If you want to change this, right click on the solution file in Visual Studio and select **Edit SampleName.csproj**. This will open the csproj-file. The first target named on this line will be used for startup/debugging in Visual Studio:

<TargetFrameworks>net45;net35;netcoreapp2.0</TargetFrameworks

### Unix-like systems

On Unix-like systems, remove the net45 and net35 targets from all projects, because they are only available on Windows. Other than that, the library and samples should compile and run fine.

1. Actually, this will only replace text in the body of the document, and not in headers, footers, end- or footnotes. So this won’t work as expected. But you can use code here. [↑](#footnote-ref-2)