Version 2.6.0.0  
egonl  
September 2025

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. 8-9-2025 23:15:03 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. This example also shows how to use streams instead of files for views and/or generated documents.

SharpDocx is inspired by Web technologies like ASP.NET and JSP. Developers familiar with those technologies should feel right at home. It supports .NET Framework 3.5-4.8 and .NET Standard 2.0. Since it supports .NET Standard 2.0 it can be used in .NET Core 3.1 and .NET 5.0 - .NET 8.0 projects as well.

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](#_Toc48598503)

[The Write method 3](#_Toc48598504)

[Conditional content 4](#_Toc48598505)

[Text block limitations 4](#_Toc48598506)

[Loops 5](#_Toc48598507)

[Nested loops 6](#_Toc48598508)

[Loops, tables and the AppendRow method 7](#_Toc48598509)

[Combining loops, text blocks and tables 8](#_Toc48598510)

[Images 9](#_Toc48598511)

[Replacing text 10](#_Toc48598512)

[Referencing assemblies and importing namespaces 11](#_Toc48598513)

[Notes 11](#_Toc48598514)

[The Map 13](#_Toc48598515)

[The SharpDocx solution 14](#_Toc48598516)

# 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 a paragraph 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.

## Text block limitations

1. You can’t use text blocks to conditionally display a part of a paragraph. It’s all or nothing. This makes the text block implementation much simpler. However, it might also give some unexpected results.

2. Text blocks can’t share paragraphs. That means that you can’t write < % } } % > to end two text blocks. Instead, use two paragraphs, each containing < % } % >.

3. Also, text blocks in else statements are at the moment not supported by SharpDocx. Instead, use another if statement.

4. Don’t mix text blocks with the AppendRow or AppendParagraph methods: it just won’t work. Instead, use the Write method to display conditional content. See also issue #25.

# Loops

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

Infobox_info_icon.emfThe value of i is **1.**i *squared* is **1**

Infobox_info_icon.emfThe value of i is **2.**i *squared* is **4**

Infobox_info_icon.emfThe value of i is **3.**i *squared* is **9**

Infobox_info_icon.emfThe value of i is **4.**i *squared* is **16**

Infobox_info_icon.emfThe value of i is **5.**i *squared* is **25**

Infobox_info_icon.emfThe value of i is **6.**i *squared* is **36**

Infobox_info_icon.emfThe value of i is **7.**i *squared* is **49**

Infobox_info_icon.emfThe value of i is **8.**i *squared* is **64**

Infobox_info_icon.emfThe value of i is **9.**i *squared* is **81**

Infobox_info_icon.emfThe value of i is **10.**i *squared* is **100**

Infobox_info_icon.emfThe value of i is **11.**i *squared* is **121**

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 ‘C:\Projects\SharpDocx\Samples\SampleProjects\Tutorial\bin\Debug\net48/../../../../../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:



New methods in SharpDocx 2.4 are ImageFromBase64 and ImageFromUrl. And if you want you can now also use your own streams using the ImageFromStream(Stream stream, int percentage = 100, string extension = null)method.

 ImageFromBase64 example.

 ImageFromUrl example.

SharpDocx supports the following image formats: bmp, gif, jpeg, png, tiff and emf.

# 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.

**Paramount and Activision Team For 'Call of Duty' Movie**Paramount and Activision are teaming up to produce a live-action Call of Duty movie, with Paramount promising the same blockbuster treatment it gave Top Gun: Maverick.  
   
David Ellison, Chairman and CE…

**32GB of RAM On Track To Become the New Majority For Gamers**Steam's August 2025 hardware survey shows 32GB RAM configurations reached 35.42% of users while 16GB systems fell to 41.67%, continuing a six-month trend that positions 32GB to become the dominant mem…

**Blizzard's 'Diablo' Devs Unionize. There's Now 3,500 Unionized Microsoft Workers**PC Gamer reports:  
  
The Diablo team is the next in line to unionize at Blizzard. Over 450 developers across multiple disciplines have voted to form a union under the Communications Workers of America (…

**Battlefield 6 Dev Apologizes For Requiring Secure Boot To Power Anti-Cheat Tools**An anonymous reader quotes a report from Ars Technica: Earlier this month, EA announced that players in its Battlefield 6 open beta on PC would have to enable Secure Boot in their Windows OS and BIOS …

**Today's Game Consoles Are Historically Overpriced**ArsTechnica: Today's video game consoles are hundreds of dollars more expensive than you'd expect based on historic pricing trends. That's according to an Ars Technica analysis of decades of pricing d…

**Microsoft Expands Xbox Cloud Gaming to Cheaper Game Pass Tiers**Microsoft is testing new Xbox Game Pass features with Insiders, letting Core and Standard subscribers stream cloud-enabled titles they own or access via subscription across more devices, including sup…

**Microsoft Re-joins Handheld Gaming Fight Against Nintendo's Switch**Gaming handhelds are becoming the industry's new battleground as Microsoft launches its ROG Xbox Ally devices October 16, chasing Nintendo Switch 2's record-breaking 5.8 million units sold in seven we…

**Sony Raises PS5 Prices by $50 Across All Models in US**Sony will increase PlayStation 5 console prices by $50 across all models in the United States starting August 21. The standard PS5 rises to $550, the Digital Edition to $500, and the PS5 Pro to $750. …

**More Game Workers at Microsoft's 'Blizzard' Join a Union**This week workers on Blizzard's "Story and Franchise Development" team "strongly voted" to join America's largest communications and media labor union, the Communications Workers of America.   
  
From th…

**Ex-PlayStation Boss Says Game Subscription Turns Developers Into 'Wage Slaves'**Former Sony Worldwide Studios chairman Shawn Layden criticized subscription gaming services like Xbox Game Pass, arguing that developers working under such models become "wage slaves." Speaking in a r…

**Millions Flock To Grow Virtual Gardens In Viral Roblox Game**Grow a Garden, a Roblox game created by a 16-year-old in just a few days, has shattered records for the most concurrent players in gaming history, surpassing Fortnite with over 21.6 million concurrent…

**Digital Foundry, the Most Trusted Name in Game Console Analysis, is Going Independent**Digital Foundry, the gaming hardware analysis publication known for its technical console breakdowns, has separated from IGN ownership as of today, with founder Richard Leadbetter purchasing the outle…

**Electronic Arts Tries (Once More) To End Its Football Addiction**Electronic Arts faces a familiar challenge as it prepares to launch Battlefield 6 on October 10: breaking its dependence on the FIFA franchise, now called EA Sports FC, which drives roughly 70% of com…

**Call of Duty's Anti-Cheat Will Require TPM 2.0 and Secure Boot for PC Players**Activision will require PC players of Call of Duty: Black Ops 7 to enable Trusted Platform Module 2.0 and Windows Secure Boot when the game launches later this year. The company begins testing these a…

**Disney Struggles With How to Use AI - While Retaining Copyrights and Avoiding Legal Issues**Disney "cloned" Dwayne Johnson when filming a live-action Moana, reports the Wall Street Journal, using an AI process that they were ultimately afraid to use:  
  
Under the plan they devised, Johnson's s…

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.

## Notes

SharpDocx will automatically reference the calling assembly. So if the view model is declared in the calling assembly, you can use that model in your document without explicitly referencing that assembly. However, if the view model is defined in another assembly, you need to explicitly reference it. If you don't, you'll get compilation errors like:

Line 26: error CS0012: The type 'ClassLibrary1.Models.Country' is defined in an assembly that is not referenced. You must add a reference to assembly 'ClassLibrary1, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null'.

The simplest way to add a reference to ClassLibrary1 is by using an Assembly-directive in your document:

< %@ Assembly Name="ClassLibrary1" % >

Or, if you're using .NET Core, you might want to use:

< %@ Assembly Name="~/ClassLibrary1" % >

The tilde represents the directory that contains SharpDocx.dll. Use it when you get errors like:

System.IO.FileNotFoundException: Could not find file 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\2.0.9\ClassLibrary1.dll'.

Another way to add references and namespaces is by defining your own SharpDocx document subclass. See the Inheritance 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 2.6.0.0  
egonl  
September 2025  
Version 2.6.0.0  
egonl  
September 2025  
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. 8-9-2025 23:15:03 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 built for .NET Framework 4.8 and .NET 6.0. The.NET 6.0 build will use the .NET Standard 2.0 version of SharpDocx.

The samples will by default run in .NET Framework 4.8 in Visual Studio 2022. If you want to change this, right click on the project file in and select **Edit Project File**. This will open the csproj file. The first target named on this line will be used for startup/debugging in Visual Studio:

<TargetFrameworks>net48;net6.0</TargetFrameworks>

Depending on the SDKs you have installed, you can choose between net35, net40, net45, net46, net47, net48, netstandard2.0, netcoreapp3.1, net5.0, net6.0 and net7.0.

### Linux and Mac

First clone the SharpDocx repository:

git clone https://github.com/egonl/SharpDocx

cd SharpDocx

Now you can build and run the Tutorial sample.

dotnet build SharpDocx.sln

dotnet Samples/SampleProjects/Tutorial/bin/Debug/net6.0/Tutorial.dll

If you want you can remove the net48 target from all projects on Unix-like systems, because they will produce Windows executables.

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)