Version <% { var sharpDocxAssembly = typeof(DocumentBase).Assembly;  
var fvi = System.Diagnostics.FileVersionInfo.GetVersionInfo(sharpDocxAssembly.Location);  
Write(fvi.FileVersion); } %>  
egonl  
January 2018

SharpDocx

# Summary

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

First you create a view in Word. A view is a Word document which also contains C# code. Code can be inserted anywhere: <%= 3\*8 %> would insert 24.

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.

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

[Summary 1](#_Toc502494466)

[Contents 2](#_Toc502494467)

[The basics 3](#_Toc502494468)

[Write method 3](#_Toc502494469)

[Conditional content 3](#_Toc502494470)

[Loops 4](#_Toc502494471)

[Tables 5](#_Toc502494472)

[Images 6](#_Toc502494473)

[Replacing text 7](#_Toc502494474)

[Referencing assemblies and importing namespaces 8](#_Toc502494475)

[The Map 9](#_Toc502494476)

# The basics

At any point in the text you can insert C# statements. Like right <% var i = 1; %>here.

The result looks like this:



## Write method

If you want to display the value of i, you can use the Write method. Right now, i is <% Write(i); %>.

There’s also a shorthand notation for the Write method: i is still <%= i %>.

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

<%= "This paragraph\ncontains two\nline breaks" %>.

## Conditional content

You can use an if statement to display conditional content.

<% if (true) { Write("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 two code blocks and place the text between curly brackets, like this.

<% if (true) { %>This **will** *also* be displayed.<% } %>

If you want, you can span multiple elements. E.g.

<% if (DateTime.Now.Second % 2 == 0) { %>

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.

<% } %>

## Loops

If you want to add something in a loop, you should do so programmatically. This example does loop, but doesn’t add anything to the document:

<% for (i = 0; i < 10; ++i) { %>

The value of i is **<%= i %>.**

<% } %>

In the next loop we’ll call the AppendParagraph method, which *will* insert new paragraphs:

<% for (i = 0; i < 10; ++i) { %>

The value of i is **<%= i %>.**<% AppendParagraph(); %>

<% } %>

## Tables

Design your tables in Word, and create rows using the AppendRow method.

<% for (i = 1; i <= 10; ++i) { %>

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i \* 1 | i \* 2 | i \* 3 | i \* 4 | i \* 5 | i \* 6 | i \* 7 | i \* 8 | i \* 9 | i \* 10 |
| <%= i %> | <%= i \* 2 %> | <%= i \* 3 %> | <%= i \* 4 %> | <%= i \* 5 %> | <%= i \*6 %> | <%= i \* 7 %> | <%= i \* 8 %> | <%= i \* 9 %> | <%= i \* 10 %> <% AppendRow(); %> |

<% } %>

## Images

Insert images using the Image method.

<% Image("test1.png"); %>

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 ‘<%= ImageDirectory %>’.

The Image method accepts a second optional parameter that specifies the relative size of the image. Here’s <% Image("test1.png", 15); %> at 15%.

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

<% Image("test2.png"); %>

## Replacing text

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

<% Replace("{text to replace}", "replaced text"); %>

This will replace *all* occurrences of the specified string.

Here’s the **{text to replace}**. And here’s some more {text *to* **replace**}.

<% /\* Replace("e", "é"); \*/ %>

## 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.

<%@ Assembly Name="System.Xml" %>  
<%@ Assembly Name="System.Xml.Linq" %>

Import namespaces with the Import directive.

<%@ Import Namespace="System.Xml.Linq" %>

In C# you would write:

using System.Xml.Linq;

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

<% try  
{  
 var atom = XDocument.Load("http://rss.slashdot.org/Slashdot/slashdotMainatom");

foreach (var entry in atom.Descendants("{http://www.w3.org/2005/Atom}entry"))  
 { %>

**<%=** **entry.Element("{http://www.w3.org/2005/Atom}title").Value %>**<%= entry.Element("{http://www.w3.org/2005/Atom}summary").Value.Substring(0,200) %>…<% AppendParagraph(); %>

<% }  
}  
catch (Exception ex)  
{   
 Write("Error: " + ex.Message);   
} %>

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:

<%= Map.Text.Substring(0,500) %> …

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