egonl  
Version 0.1  
August 2017

SharpDocx

# Summary

SharpDocx is a library to programmatically create Word documents. It’s inspired on Web technologies like ASP.NET. Developers familiar with classic ASP.NET should feel right at home.

With SharpDocx, the developer first creates a view. This is a Word document which also contains C# code. This is usually done in Microsoft Word.

Next, the developer programmatically creates new documents based on this view. The views can optionally be supplied with a user defined model. This is usually done in Visual Studio using C#.

Out of the box SharpDocx supports most standard scenario’s, like inserting text, tables and images. If something more specific is required, developers can extend views in order to implement those requirements.

# Contents

[Summary 1](#_Toc500358764)

[Contents 2](#_Toc500358765)

[The basics 3](#_Toc500358766)

[Write method 3](#_Toc500358767)

[Conditional content 3](#_Toc500358768)

[Loops 3](#_Toc500358769)

[Replacing text 5](#_Toc500358770)

[The map 5](#_Toc500358771)

# 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 %>.

## 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. Jun 22, 2015

<% } %>

## Loops

If you want to add something in a loop, you should do so programatically. 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 CreateParagraph method, which *will* insert new paragraphs:

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

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

<% } %>

## Tables

<% 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 %> <% CreateRow(); %> |

<% } %>

## Images

<% Image("test1.png", 30); %>

<% Image("test1.png", 50); %>

<% Image("test1.png", 100); %>

An old version of SharpDocx in Visual Studio: <% Image("test2.png", 50); %>

## Replacing text

If you want to replace text, you can use the Map.Replace method <% Map.Replace("{text to replace}", "replaced text"); %>. It will replace all occurences of the specified string.

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

## The Map

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

<%= Map.Text %>