



Word automation using C#

Word Automation through C# is all about programmatically generating the Word Document using C# code. Almost all of the tasks which we perform on word 2003 can be done programmatically using C# or VB.



Amrish Deep Ravidas

Oct 10 2018

8

54

963.1k

[Download Free .NET & JAVA Files API](#)

1. Development Tools Used

- Microsoft Visual Studio 2005
- Microsoft Word 2003
- Programming Language: C#

2. Word Automation using C#

Word Automation through C# is all about programmatically generating the Word Document using C# code. Working on Word is considered to be straightforward, but doing the same programmatically gets a little intricate. Word automation almost completely involves working with objects and reference types. Almost all of the tasks which we perform on word 2003 can be done programmatically using C# or VB. Tasks like Inserting Table of Contents, Linking documents, Mail Merge, Inserting Documents, Embedding documents, inserting pictures, watermark... etc can all be done programmatically.

3. Setting Up Work Environment

Starting off, the first step is to include the Word dll's to the Solution. This can be done by right clicking the Reference Folder in the Solution explorer of the project and select Add Reference.

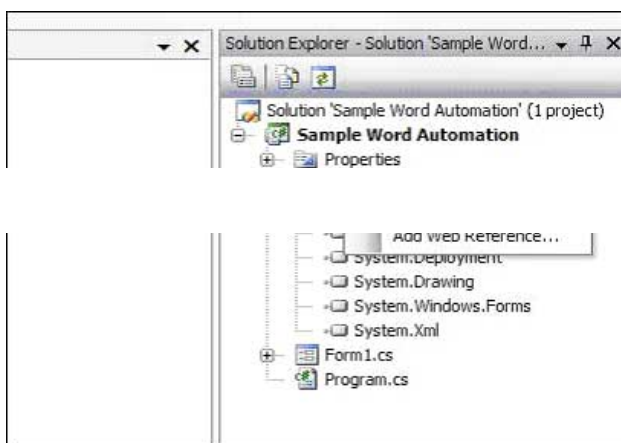


Figure 1.

Browse Through the available COM objects and Select Microsoft Office 11.0 Object Library & Microsoft Word 11.0 Object Library. This DLL has all the methods which we do to perform the automation.

Also include "using Microsoft.Office;" in the *Namespaces used*.

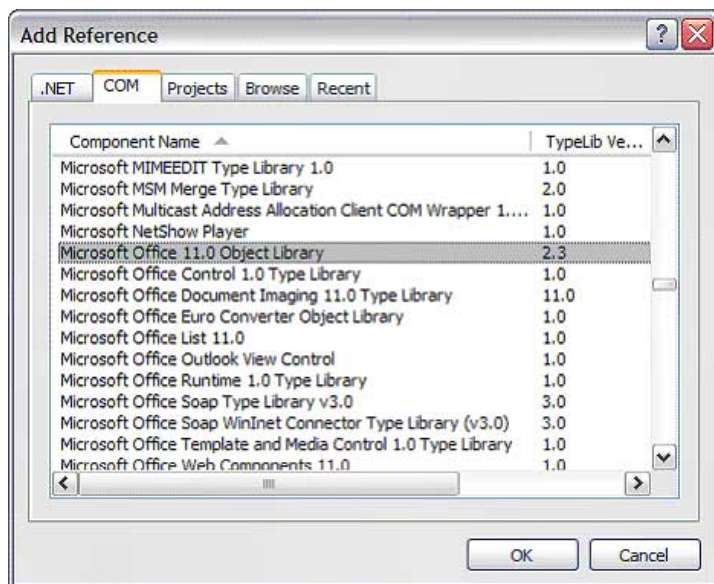
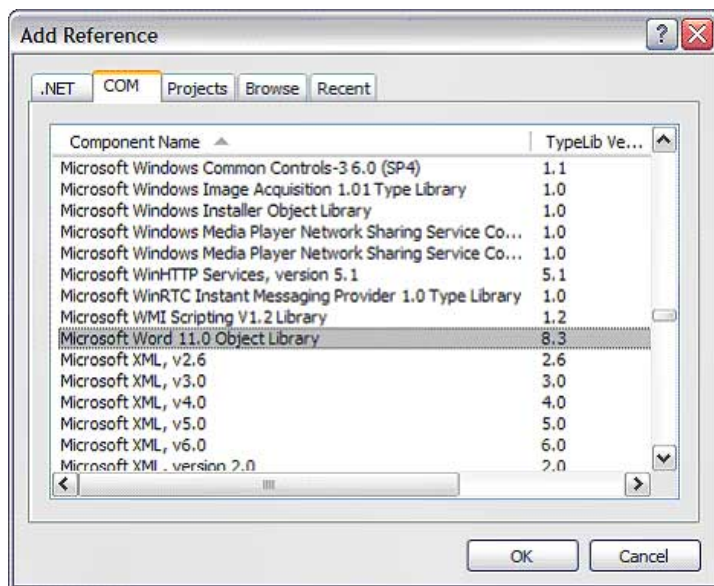
[ASK A QUESTION](#)
[CONTRIBUTE](#)


Figure 2.



4. Objects Used in Automation

All the methods used Word automation is derived either from **Word.Application** or **Word.Document** class.

Let's consider that we want to create a document using the Word Application, we might end up doing the following steps,

1. Open Word Application. (Opening Word Application creates a new document by default, but in Automation, we need to manually add a document)
2. Add a New document.
3. Edit the document.
4. Save it.

This represents in Word Application without any new document loaded in it. This is like the base class which is needed to create a new document. Creating a new instance of Word.Application can be visualized as below.

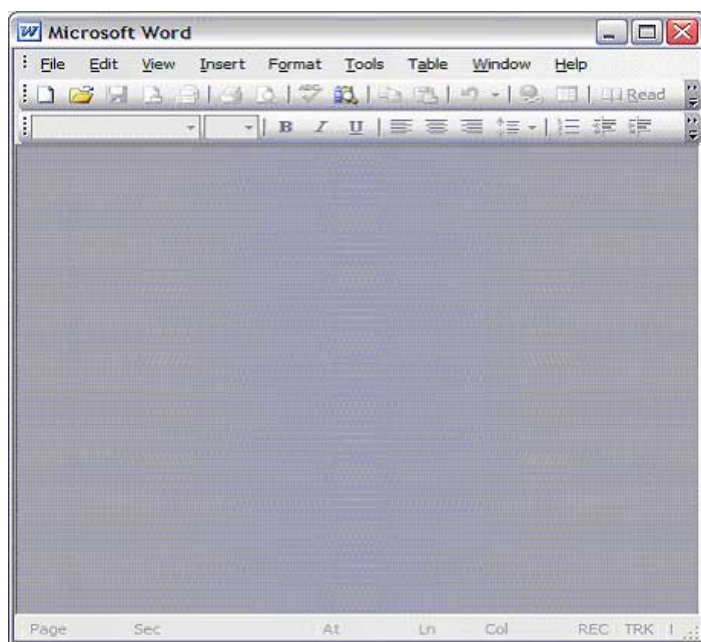


Figure 4.

4.2 Word.Document

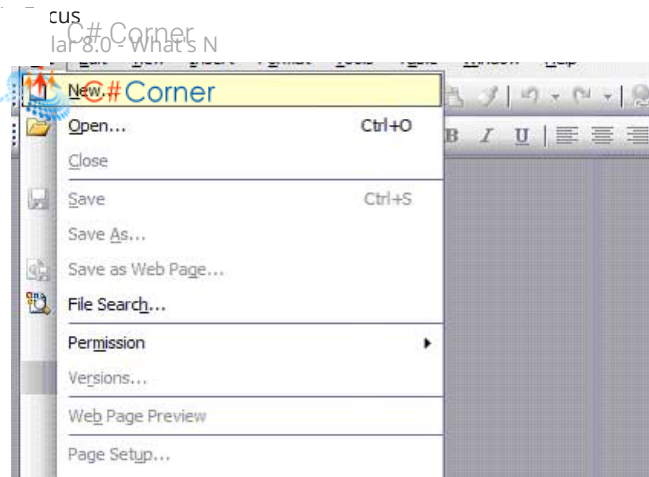
If we need to add a new document file, first we have to create an instance of the **Word.Document** object and then add it to the **Word.Application**.

```

01. //OBJECT OF MISSING "NULL VALUE"
02. Object oMissing = System.Reflection.Missing.Value();
03. //OBJECTS OF FALSE AND TRUE
04. Object oTrue = true;
05. Object oFalse = false;
06. //CREATING OBJECTS OF WORD AND DOCUMENT
07. Word.Application oWord = new Word.Application();
08. Word.Document oWordDoc = new Word.Document();
09. //MAKING THE APPLICATION VISIBLE
10. oWord.Visible = true;
11. //ADDING A NEW DOCUMENT TO THE APPLICATION

```

This triggers the following operation in the Word Application



ASK A QUESTION

CONTRIBUTE

Figure 5.

Approaches to Perform Automation

1. We can either have a base template (.dot) file and open the base template file and work on it.
2. We can otherwise build a word document from scratch.

4.3 Standard Input Parameters

Most of the methods have input parameters which are of reference type, and the values are mostly true, false or missing (null). In automation it makes sense as to why most of the input parameters are of reference types; it might be because of the fact that most of the methods a multitude of input parameters (many have more than 10 input parameters) and their value is going to be either true, false or missing in most of the cases. So instead of supplying the same input parameter ten times, we can make all the input parameters point to the location same single variable in them memory.

4.3.1 Range Object

While we work on Word Application, if we want to type some text in the 11th line, then we manually take the cursor and click it on the required line and then start typing. In order to do the same task, we use the Range variable in C#. The range variable of the **Word.Document** object represents the location of the cursor on the current document.

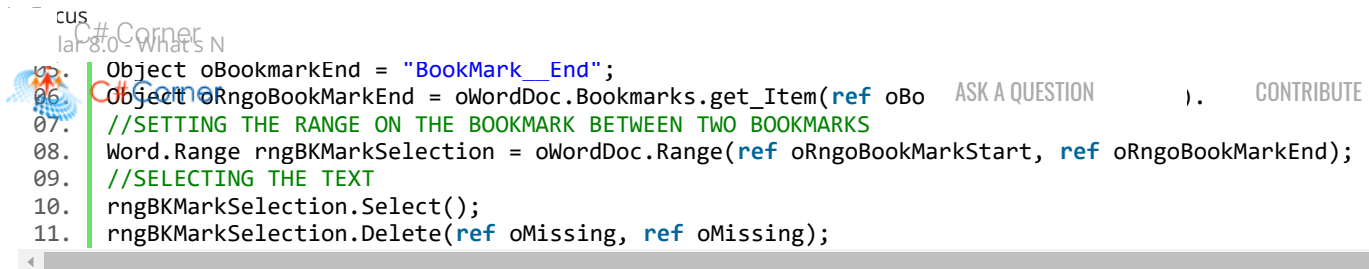
There are many possible ways to point to a specific location on a document. I had extensively used the Bookmarks locators as I work on Automation using a base template. In this approach, we insert Bookmarks on the base template and we programmatically locate those Bookmarks, set the range on them and insert text or documents at that specific location. There are also many other possible ways to set the range.

```
01. //SETTING THE RANGE ON THE BOOKMARK
02. Object oBookMarkName = "My_Inserted_Bookmark_On_Template";
03. Word.Range wrdRange = oWordDoc.Bookmarks.get_Item(ref oBookMarkName).Range.Select();
```

While working on word, we select a range of text by clicking and dragging the mouse pointer across contents in the document to select it. The contents can be text, formatted text, tables or any other item in the document. We programmatically represent the same by using the Selection Object derived from **Word.Selection**. In the previous range example, we locate a bookmark and set the range on that specific bookmark and we select it. Now the selection object represents that specific location. It's like placing the cursor on that specific bookmark location on the document. The selection across text can be done by selecting a range of text in between two ranges. Then the selected range can be copied, deleted or formatted.

4.3.3 Selecting Between Bookmarks

```
01. //BOOK MARK FOR START OF SELECTION
02. Object oBookmarkStart = "BookMark__Start";
```



```

05. Object oBookmarkEnd = "BookMark_End";
06. Object oRngoBookMarkEnd = oWordDoc.Bookmarks.get_Item(ref oBo
07. //SETTING THE RANGE ON THE BOOKMARK BETWEEN TWO BOOKMARKS
08. Word.Range rngBKMarkSelection = oWordDoc.Range(ref oRngoBookMarkStart, ref oRngoBookMarkEnd);
09. //SELECTING THE TEXT
10. rngBKMarkSelection.Select();
11. rngBKMarkSelection.Delete(ref oMissing, ref oMissing);

```

5. Automation using a Base Template

The base template file method is preferable as it gives us much more flexibility in performing the automation and it comes very handy for performing *Mail Merge*.

In the base template method, when we call the **Documents.Add** method of the Application object, we give the path of the .dot file.

```

01. //THE LOCATION OF THE TEMPLATE FILE ON THE MACHINE
02. Object oTemplatePath = "C:\\Program Files\\MyTemplate.dot";
03. //ADDING A NEW DOCUMENT FROM A TEMPLATE
04. oWordDoc = oWord.Documents.Add(ref oTemplatePath, ref oMissing, ref oMissing, ref oMissing);

```

Now .dot file is opened and when we save the generated document, we save it as a new file.

6. Mail Merge

Mail merge is a useful tool in scenarios where we want to randomly generate alike documents where just a few fields change. For instance in a pay slip which has a base template and just the employee name, number and pay details needs to change for each employee. Now we can have a base template which is a word file saved as Document Template file.

In the .dot file, insert a Mail Merge Field manually by placing the cursor in the required position and **Insert -> Field**, and in Field Names, select **"MergeField"**, now the Mail merged field would be represented by <<FieldName>>. The template can be like

Contact Information

For further information and discussions, please contact:

Name: <<CIFLName>>

Address: <<CIAddress>>

Phone: <<CIPhW>> (Work)

<<CIPhM>>(Cell)

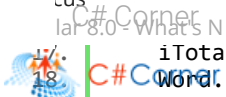
Fax: <<CIFax>>

Email <<CIMail>>

```

01. //OBJECT OF MISSING "NULL VALUE"
02. Object oMissing = System.Reflection.Missing.Value();
03. //OBJECTS OF FALSE AND TRUE
04. Object oTrue = true;
05. Object oFalse = false;
06. //CREATING OBJECTS OF WORD AND DOCUMENT
07. Word.Application oWord = new Word.Application();
08. Word.Document oWordDoc = new Word.Document();
09. //SETTING THE VISIBILITY TO TRUE
10. oWord.Visible = true;
11. //THE LOCATION OF THE TEMPLATE FILE ON THE MACHINE
12. Object oTemplatePath = "C:\\Program Files\\MyTemplate.dot";
13. //ADDING A NEW DOCUMENT FROM A TEMPLATE
14. oWordDoc = oWord.Documents.Add(ref oTemplatePath, ref oMissing, ref oMissing, ref oMissing);

```



```

17. iTotalFields++;
18. Word.Range rngFieldCode = myMergeField.Code;
19. String fieldText = rngFieldCode.Text;
20. // ONLY GETTING THE MAILMERGE FIELDS
21. if (fieldText.StartsWith(" MERGEFIELD"))
22. {
23.     // THE TEXT COMES IN THE FORMAT OF
24.     // MERGEFIELD MyFieldName \[* MERGEFORMAT
25.     // THIS HAS TO BE EDITED TO GET ONLY THE FIELDNAME "MyFieldName"
26.     Int32 endMerge = fieldText.IndexOf("\\");
27.     Int32 fieldNameLength = fieldText.Length - endMerge;
28.     String fieldName = fieldText.Substring(11, endMerge - 11);
29.     // GIVES THE FIELDNAMES AS THE USER HAD ENTERED IN .dot FILE
30.     fieldName = fieldName.Trim();
31.     // **** FIELD REPLACEMENT IMPLEMENTATION GOES HERE ****//
32.     // THE PROGRAMMER CAN HAVE HIS OWN IMPLEMENTATIONS HERE
33.     if (fieldName == "MyField")
34.     {
35.         myMergeField.Select();
36.         oWord.Selection.TypeText("This Text Replaces the Field in the Template");
37.     }
38. }
39. }

```

ASK A QUESTION CONTRIBUTE

There is one other method for replacing the [Merge Fields which is mentioned in msdn](#), which uses a rather memory hungry approach. In that method a separate document is opened and it is inserted with a table which has first row as the Mail Merge Field Name and the second row as the replacement value, then the value from the table is matched with that of the original document and replacement occurs and the second document is purged.

7. Embedding a Document

Embedding a document is done through the application by

Insert-> Object-> Create from file-> Select the File-> Display as Icon. This embeds the file in the selected location as an icon and the user can double click on the icon to open the file. The same can be done through automation.

The range supposed to set at the required place and the same has to be selected (range can be set by any of the means mentioned above). Now with the selection, the file can be embedded.

```

01. //ICON LABEL CAN BE THE NAME OF THE FILE,
02. //ITS THE NAME DISPLAYED BESIDES THE EMBEDDED DOCUMENT
03. Object oIconLabel = "File Name";
04. //INCASE WE NEED THE EMBEDDED DOCUMENT TO BE DISPLAYED AS A SPECIFIC ICON,
05. //WE NEED TO SPECIFY THE LOCATION OF THE ICON FILE
06. //ELSE SET IT TO oMissing VALUE
07. Object oIconFileName = "C:\\Document and Settings\\IconFile.ico";
08. //THE BOOKMARK WHERE THE FILE NEEDS TO BE EMBEDDED
09. Object oBookMark = "My_Custom_BookMark";
10. //THE LOCATION OF THE FILE

11.
12.
13. Object oClassType = word.DocumentClass;
14. Object oTrue = true;
15. Object oFalse = false;
16. Object oMissing = System.Reflection.Missing.Value;
17. //METHOD TO EMBED THE DOCUMENT
18. oWordDoc.Bookmarks.get_Item(ref oBookMark).Range.InlineShapes.AddOLEObject(
19.     ref oClassType,ref oFileDesignInfo,ref oFalse, ref oTrue, ref oIconFileName,
20.     ref oMissing,ref oIconLabel, ref oMissing);

```

8. Inserting a Document File

Contents of a Word documents can also be inserted into the current document from the application by doing the following.

Insert -> File -> Select the File. This extracts the contents from the selected file and inserts it into the current document.



```
01. //THE LOCATION OF THE FILE
02. String oFilePath = "C:\\Document and Settings\\somefile.doc";
03. oWordDoc.Bookmarks.get_Item(ref oBookmark).Range.InsertFile(oFilePath,ref oMissing, ref oFalse, ref
```

9. Including Water Marks/Pictures in the Document Background

Including watermarks is one other important feature for any official documents as the watermark may have the company's logo, draft logo or any other picture/text. This is useful when we want a picture or some text to be present throughout the document in the background.

We insert a watermark in the application by performing the following tasks.

Format -> Background -> Printed Watermarks

The same can also be done programmatically; moreover as we manually define the values like the angle of tilt and actual location of the watermark, we have more flexibility in defining the exact location of the watermark.

9.1 Embedding Pictures in Document Header

```
01. //EMBEDDING LOGOS IN THE DOCUMENT
02. //SETTING FOCUES ON THE PAGE HEADER TO EMBED THE WATERMARK
03. oWord.ActiveWindow.ActivePane.View.SeekView = Word.WdSeekView.wdSeekCurrentPageHeader;
04. //THE LOGO IS ASSIGNED TO A SHAPE OBJECT SO THAT WE CAN USE ALL THE
05. //SHAPE FORMATTING OPTIONS PRESENT FOR THE SHAPE OBJECT
06. Word.Shape logoCustom = null;
07. //THE PATH OF THE LOGO FILE TO BE EMBEDDED IN THE HEADER
08. String logoPath = "C:\\Document and Settings\\MyLogo.jpg";
09. logoCustom = oWord.Selection.HeaderFooter.Shapes.AddPicture(logoPath,
10.     ref oFalse, ref oTrue, ref oMissing, ref oMissing, ref oMissing, ref oMissing, ref oMissing);
11. logoCustom.Select(ref oMissing);
12. logoCustom.Name = "CustomLogo";
13. logoCustom.Left = (float)Word.WdShapePosition.wdShapeLeft;
14. //SETTING FOCUES BACK TO DOCUMENT
15. oWord.ActiveWindow.ActivePane.View.SeekView = Word.WdSeekView.wdSeekMainDocument;
```

9.2 Inserting Text in the Centre of the Document as Water Mark

```
01. //THE LOGO IS ASSIGNED TO A SHAPE OBJECT SO THAT WE CAN USE ALL THE
02. //SHAPE FORMATTING OPTIONS PRESENT FOR THE SHAPE OBJECT
03. Word.Shape logoWatermark = null;
04. //INCLUDING THE TEXT WATER MARK TO THE DOCUMENT
05. logoWatermark = oWord.Selection.HeaderFooter.Shapes.AddTextEffect(
06.     Microsoft.Office.Core.MsoPresetTextEffect.msoTextEffect1,
07.     "Enter The Text Here", "Arial", (float)60,
08.     Microsoft.Office.Core.MsoTriState.msoTrue,
09.     Microsoft.Office.Core.MsoTriState.msoFalse,
10.     (float)Word.WdShapePosition.wdShapeCenter, (float)Word.WdShapePosition.wdShapeCenter,
11.     (float)Word.WdShapePosition.wdShapeCenter, (float)Word.WdShapePosition.wdShapeCenter);
12. logoWatermark.Fill.Visible = Microsoft.Office.Core.MsoTriState.msoTrue;
13. logoWatermark.Line.Visible = Microsoft.Office.Core.MsoTriState.msoFalse;
14. logoWatermark.Fill.Solid();
15. logoWatermark.Fill.ForeColor.RGB = (Int32)Word.WdColor.wdColorGray30;
16. logoWatermark.RelativeHorizontalPosition = Word.WdRelativeHorizontalPosition.wdRelativeHorizontalPositionCenter;
17. logoWatermark.RelativeVerticalPosition = Word.WdRelativeVerticalPosition.wdRelativeVerticalPositionCenter;
18. logoWatermark.Left = (float)Word.WdShapePosition.wdShapeCenter;
19. logoWatermark.Top = (float)Word.WdShapePosition.wdShapeCenter;
20. logoWatermark.Height = oWord.InchesToPoints(2.4f);
21. logoWatermark.Width = oWord.InchesToPoints(6f);
22. //SETTING FOCUES BACK TO DOCUMENT
23. oWord.ActiveWindow.ActivePane.View.SeekView = Word.WdSeekView.wdSeekMainDocument;
```

```

01. //INSERTING TEXT IN THE CENTRE RIGHT, TILTED AT 90 DEGREES
02. Word.Shape midRightText;
03. midRightText = oWord.Selection.HeaderFooter.Shapes.AddTextEffect(
04.     Microsoft.Office.Core.MsoPresetTextEffect.msoTextEffect1,
05.     "Text Goes Here", "Arial", (float)10,
06.     Microsoft.Office.Core.MsoTriState.msoTrue,
07.     Microsoft.Office.Core.MsoTriState.msoFalse,
08.     0, 0, ref oMissing);
09. //FORMATTING THE SECURITY CLASSIFICATION TEXT
10. midRightText.Select(ref oMissing);
11. midRightText.Name = "PowerPlusWaterMarkObject2";
12. midRightText.Fill.Visible = Microsoft.Office.Core.MsoTriState.msoTrue;
13. midRightText.Line.Visible = Microsoft.Office.Core.MsoTriState.msoFalse;
14. midRightText.Fill.Solid();
15. midRightText.Fill.ForeColor.RGB = (int)Word.WdColor.wdColorGray375;
16. //MAKING THE TEXT VERTICAL & ALIGNING
17. midRightText.Rotation = (float)90;
18. midRightText.RelativeHorizontalPosition =
19.     Word.WdRelativeHorizontalPosition.wdRelativeHorizontalPositionMargin;
20. midRightText.RelativeVerticalPosition =
21.     Word.WdRelativeVerticalPosition.wdRelativeVerticalPositionMargin;
22. midRightText.Top = (float)Word.WdShapePosition.wdShapeCenter;
23. midRightText.Left = (float)480;

```

10. Including Page Numbers in Page Footer

Including auto-generated page numbers in the Footer is yet another useful feature which can be simulated in the code.

```

01. //SETTING THE FOCUES ON THE PAGE FOOTER
02. oWord.ActiveWindow.ActivePane.View.SeekView = Word.WdSeekView.wdSeekCurrentPageFooter;
03. //ENTERING A PARAGRAPH BREAK "ENTER"
04. oWord.Selection.TypeParagraph();
05. String docNumber = "1";
06. String revisionNumber = "0";
07. //INSERTING THE PAGE NUMBERS CENTRALLY ALIGNED IN THE PAGE FOOTER
08. oWord.Selection.Paragraphs.Alignment = Word.WdParagraphAlignment.wdAlignParagraphLeft;
09. oWord.ActiveWindow.Selection.Font.Name = "Arial";
10. oWord.ActiveWindow.Selection.Font.Size = 8;
11. oWord.ActiveWindow.Selection.TypeText("Document #: " + docNumber + " - Revision #: " + revisionNumber);
12. //INSERTING TAB CHARACTERS
13. oWord.ActiveWindow.Selection.TypeText("\t");
14. oWord.ActiveWindow.Selection.TypeText("\t");
15. oWord.ActiveWindow.Selection.TypeText("Page ");
16. Object currentPage = Word.WdFieldType.wdFieldPage;
17. oWord.ActiveWindow.Selection.Fields.Add(oWord.Selection.Range, ref currentPage, ref oMissing, ref oMissing);
18. oWord.ActiveWindow.Selection.TypeText(" of ");
19. Object totalPages = Word.WdFieldType.wdFieldNumPages;
20. oWord.ActiveWindow.Selection.Fields.Add(oWord.Selection.Range, ref totalPages, ref oMissing, ref oMissing);
21. //SETTING FOCUES BACK TO DOCUMENT
22. oWord.ActiveWindow.ActivePane.View.SeekView = Word.WdSeekView.wdSeekMainDocument;

```

11.1 Paragraph Break

This is equivalent to hitting the enter button in the document.

```

01. //ENTERING A PARAGRAPH BREAK "ENTER"
02. oWord.Selection.TypeParagraph();

```

11.2 Text Formatting Option

All the text formatting options available in the Word Application can also be replicated through automation.

```

01. //OTHER COMMONLY USED FORMATTING OPTIONS

```




```
oWord.Selection.Font.Italic = 1;
```

```
oWord.Selection.Font.Underline = Word.WdUnderline.wdUnderline
```

[ASK A QUESTION](#)
[CONTRIBUTE](#)

11.3 Clear Formatting

When the Formatting is applied to a selection, then the same formatting gets carried on to the next lines, in order to clear the formatting, the next line needs to be selected and ClearFormatting() method needs to be called.

```
01. //CLEARING THE FORMATTING
02. oWord.Selection.ClearFormatting();
```

12. Table of Contents

Table of Contents is very handy when it comes to official documents or some technical papers which span across many pages. Table of contents can be inserted and updated on the fly as the document gets built.

For the Table of Contents to get auto generated without any hassles, it is vital that the Headings, Sub-Headings and the Body text have their respective attributes set. When we work on the application, the values get set by themselves, we only need to edit if required. But while programming its mandatory that we set the values in the code in order to prevent any anomalies when the Table of Contents gets updated.

Below is an example of a document which was programmatically generated.

Table of Contents	
Sample Header 2	1
Sample Header 3	1

Sample Header 2

Body Message for "Sample Header 2"

Sample Header 3

Body Message for "Sample Header 3"

C:\Documents and Settings\sysadmin.BL

Figure 6.

It is apparent that the Header 2 and Header 3 and Body are formatted differently and even in the Table of Contents the

Open the above document and Outlining Tool bar, **View -> Toolbars -> Outlining**. And on moving the cursor on the Sample Header 2, we can see that the Format is Heading 2 and Outlining level is Level 2.



ASK A QUESTION

CONTRIBUTE

Figure 7.

And for Body, the Format is Normal + Arial, 10 pt and Outlining Level is Body text.

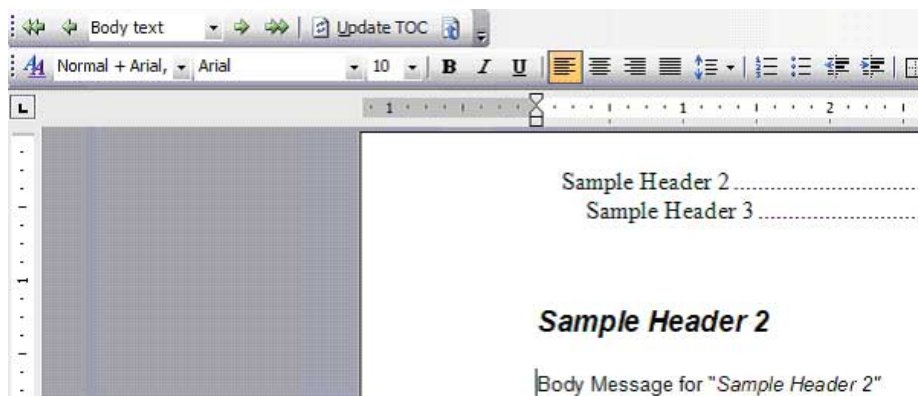


Figure 8.

The same values needs to be set programmatically for the Table of Contents to get generated.

12.1 Section Format

For setting the Format of the Selection, select the entire text (select between bookmarks like mentioned before in Selection section) and set the value

```
01. //SETTING THE FORMAT TYPE
02. //SELECT THE CONTENST TO BE FORMATTED AND SET THE VALUE
03. Object styleHeading2 = "Heading 2";
04. Object styleHeading3 = "Heading 3";
05. oWord.Selection.Range.set_Style(ref styleHeading2);
06. oWord.Selection.Range.set_Style(ref styleHeading3);
```

12.2 Outline Level

```
01. //SETTING THE OUTLINE LEVEL
02. //SELECT THE CONTENTS WHOSE OUTLINE LEVEL NEEDS TO BE CHANGED AND
03. //SET THE VALUE
04. oWord.Selection.Paragraphs.OutlineLevel = Word.WdOutlineLevel.wdOutlineLevel2;
05. oWord.Selection.Paragraphs.OutlineLevel = Word.WdOutlineLevel.wdOutlineLevel3;
06. oWord.Selection.Paragraphs.OutlineLevel = Word.WdOutlineLevel.wdOutlineLevelBodyText;
```

12.3: Inserting Table of Contents

Once the *Outline* Levels & Section Style are set, the Table of Contents can be inserted programmatically and the page numbers gets populated automatically based on the Outline Levels & Section Style set by the user. ([Also refer this MSDN Link](#))

```
01. // NAME OF THE BOOKMARK IN THE DOCUMENT (.dot Template) WHERE TABLE OF
```

```

05. // SETTING THE RANGE AT THE BOOKMARK
06. Word.Range rngTOC = oWordDoc.Bookmarks.get_Item(ref oBookmark);
07. // SELECTING THE SET RANGE
08. rngTOC.Select();
09. // INCLUDING THE TABLE OF CONTENTS
10. Object oUpperHeadingLevel = "1";
11. Object oLowerHeadingLevel = "3";
12. Object oTOCTableID = "TableOfContents";
13. oWordDoc.TablesOfContents.Add(rngTOC, ref oTrue, ref oUpperHeadingLevel,
14.     ref oLowerHeadingLevel, ref oMissing, ref oTOCTableID, ref oTrue,
15.     ref oTrue, ref oMissing, ref oTrue, ref oTrue, ref oTrue);

```

[ASK A QUESTION](#)
[CONTRIBUTE](#)

12.4 Updating Table of Contents

Usually the Table of Contents is inserted in the beginning of the document generation and once all the contents are populated, the locations of the *Headings* and Sub Headings tend to change. If the Table of Contents is not updated, then its contents points to different pages. To overcome this hassle, the Table of Contents needs to be updated at the end of the Automation.

```

01. //UPDATING THE TABLE OF CONTENTS
02. oWordDoc.TablesOfContents[1].Update();
03. //UPDATING THE TABLE OF CONTENTS
04. oWordDoc.TablesOfContents[1].UpdatePageNumbers();

```

13. Saving/Closing & Re-Opening the File

13.1 Saving the File

```

01. //THE LOCATION WHERE THE FILE NEEDS TO BE SAVED
02. Object oSaveAsFile = (Object)"C:\\SampleDoc.doc";
03. oWordDoc.SaveAs(ref oSaveAsFile, ref oMissing, ref oMissing, ref oMissing,
04.     ref oMissing, ref oMissing, ref oMissing, ref oMissing, ref oMissing,
05.     ref oMissing, ref oMissing, ref oMissing, ref oMissing, ref oMissing,
06.     ref oMissing, ref oMissing);

```

13.2 Closing the File

```

01. //CLOSING THE FILE
02. oWordDoc.Close(ref oFalse, ref oMissing, ref oMissing);
03. //QUITTING THE APPLICATION
04. oWord.Quit(ref oMissing, ref oMissing, ref oMissing);

```

13.3 Re-Opening the File

The Open () method which we use in Word2003 dll might throw an exception if the client have another version of word installed in their machine. If the client has Word 2002, then he has to open a word file only by Open2002 () method. Open () method which comes for Word 2003 might through an exception in Word 2002 environment. And for Word 2000, there is a method called Onen2000 () and Onen2002 () for Office 2002 and so on. So it is wise to put the Onen () in a try-catch block as



Figure 10.

14. Tips for Word Automation to Create New Document (Non-Base Template Approach)

(Refer to this [MSDN link](#))

section of contents, then setting the range to point to the endofdoc Bookmark a again selecting the endofdoc which would be pointing to the end of the document which would now be after the two sections.

C#

CSharp

word

Word Automation



Amrish Deep Ravidas

<https://www.c-sharpcorner.com/members/amrish-deep-ravidas>

1613

1m

[View Previous Comments](#)

8

54



Type your comment here and press Enter Key (Minimum 10 characters)



Hi the code for enter text and rotating by 90 degree(section 9.2) gives object reference error. Is there any solution to this.

deep pandey

1787 29 0

Jun 17, 2019

0 0 Reply



Your Object for missing, the Missing.Value() cannot be used like a method. This needs to be updated.

Thomas Gordon

1813 2 0

Sep 26, 2017

1 0 Reply



Nice

Ramesh Palaniappan

184 10.7k 1.3m

Aug 22, 2016

2 0 Reply



Very good article....

Parvez Ahad

1789 26 0

Jul 14, 2016

2 0 Reply



Nice...

kalu singh rao

292 6.3k 76.2k

Jul 07, 2016

3 0 Reply

<http://codebeautify.org/csharpviewer> will help to format c# code.

Iris Panabaker

1811 4 0

Jan 22, 2016

3 0 Reply



cool stuff



Can the Length of each item in TOC be trimmed to say a specific number of characters,as in my c# application i have marked sections or paragraphs to be included in TOC so m finding the whole paragraph content being set in the TOC item itself.....

Sachin K

1778 37 0

Apr 02, 2015

2 0 Reply



can i save the web page as it is with its result calculated (it contains a tab panel with some text boxes in it and a image, in the word document ??

Rahul Sonawane

1812 3 515

Jul 16, 2014

3 0 Reply



Hi, Would you please help me how I can edit schema color?

Sahar S

1814 1 0

Jul 01

3 0

[ASK A QUESTION](#)[CONTRIBUTE](#)

TRENDING UP

- 01 What Is Replication In MongoDB? How To Configure Replication In MongoDB?
- 02 Repository Design Pattern In .NET CORE WEB API
- 03 What are the Most Popular Relational Databases
- 04 Automatic OTP Verification In iOS Using Xamarin.Forms
- 05 Implementing Here Map API In Angular 8
- 06 Interfaces In C# 8.0
- 07 Add Dynamic Image In PDF And Download PDF In Angular 7
- 08 Verify OTP Automatically In Android Without SMS Read Permission Using Xamarin.Forms
- 09 Top 10 Project Management Software



ASK A QUESTION

[VIEW ALL](#)
CONTRIBUTE

[About Us](#) [Contact Us](#) [Privacy Policy](#) [Terms](#) [Media Kit](#) [Sitemap](#) [Report a Bug](#) [FAQ](#) [Partners](#) [C# Tutorials](#)
[Common Interview Questions](#)

©2019 C# Corner. All contents are copyright of their authors.