FREE eBook

LEARNING openxml

Free unaffiliated eBook created from
Stack Overflow contributors.

#openxml

Table of Contents

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: openxml

It is an unofficial and free openxml ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official openxml.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with openxml

Remarks

This section provides an overview of what openxml is, and why a developer might want to use it.

It should also mention any large subjects within openxml, and link out to the related topics. Since the Documentation for openxml is new, you may need to create initial versions of those related topics.

Examples

Installation of OpenXML SDK and productivity tool on your computer

Go to the Microsoft link for the OpenXML SDK download. Click the red download button. On the next screen click the box next to OpenXMLSDKToolV25.msi and click next to begin the download.

Once the download is complete, launch the OpenXMLSDKToolV25.msi and follow the instructions on the screen.

The installer places the files in the following default directory:

```
"C:\Program Files (x86)\Open XML SDK\V2.5"
```

In this directory is a readme that explains how to use the SDK and a readme for the productivity tool.

Create a new Spreadsheet with OpenXML

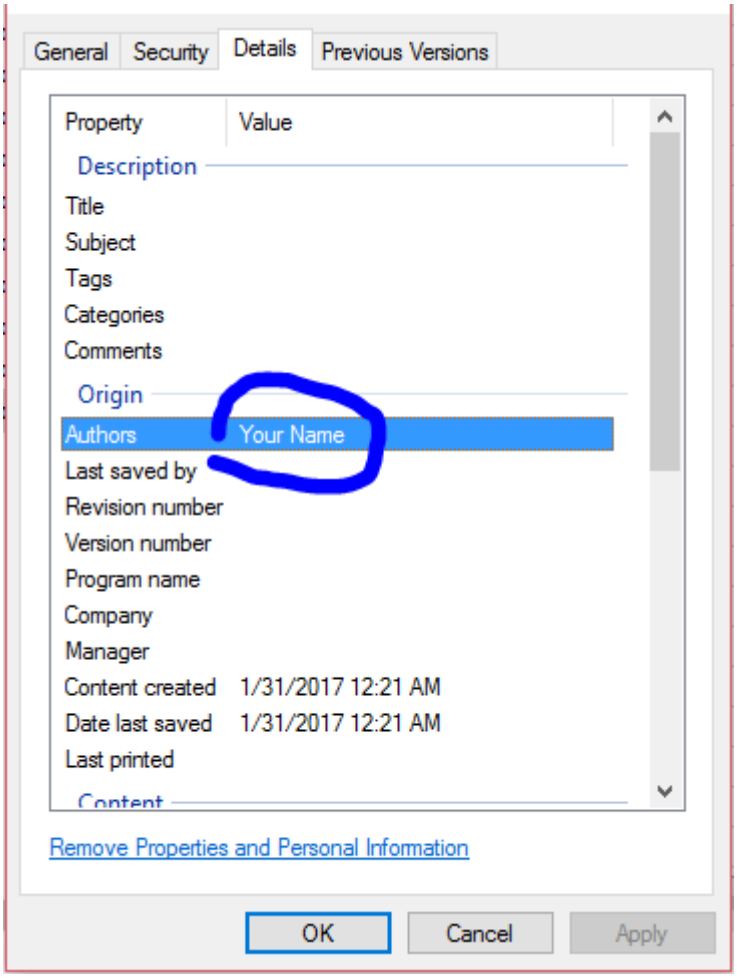
This method will create a new Excel Spreadsheet. Pass in the `fileName` which is a full file path name.

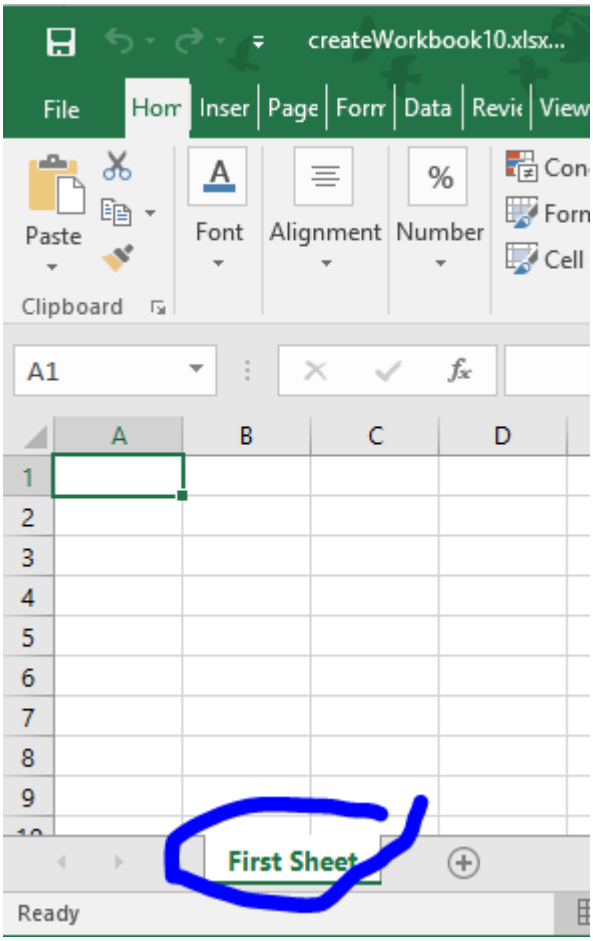
```
using DocumentFormat.OpenXml;
using DocumentFormat.OpenXml.Packaging;
using DocumentFormat.OpenXml.Spreadsheet;
using System;
....
    void Create(string fileName)
    {

        using (SpreadsheetDocument document = SpreadsheetDocument.Create(fileName,
SpreadsheetDocumentType.Workbook))
        {
            var relationshipId = "rId1";

            //build Workbook Part
            var workbookPart = document.AddWorkbookPart();
            var workbook = new Workbook();
            var sheets = new Sheets();
            var sheet1 = new Sheet() { Name = "First Sheet", SheetId = 1, Id = relationshipId
```

```
};
            sheets.Append(sheet1);
            workbook.Append(sheets);
            workbookPart.Workbook = workbook;

            //build Worksheet Part
            var workSheetPart = workbookPart.AddNewPart<WorksheetPart>(relationshipId);
            var workSheet = new Worksheet();
            workSheet.Append(new SheetData());
            workSheetPart.Worksheet = workSheet;

            //add document properties
            document.PackageProperties.Creator = "Your Name";
            document.PackageProperties.Created = DateTime.UtcNow;

        }
```

For this project, make sure you include the reference to `DocumentFormat.OpenXml`. This is located in the path specified in the Installing OpenXML Example.

The spreadsheet will be created with **Your Name** as the Author and the first Worksheet named **First Sheet**.



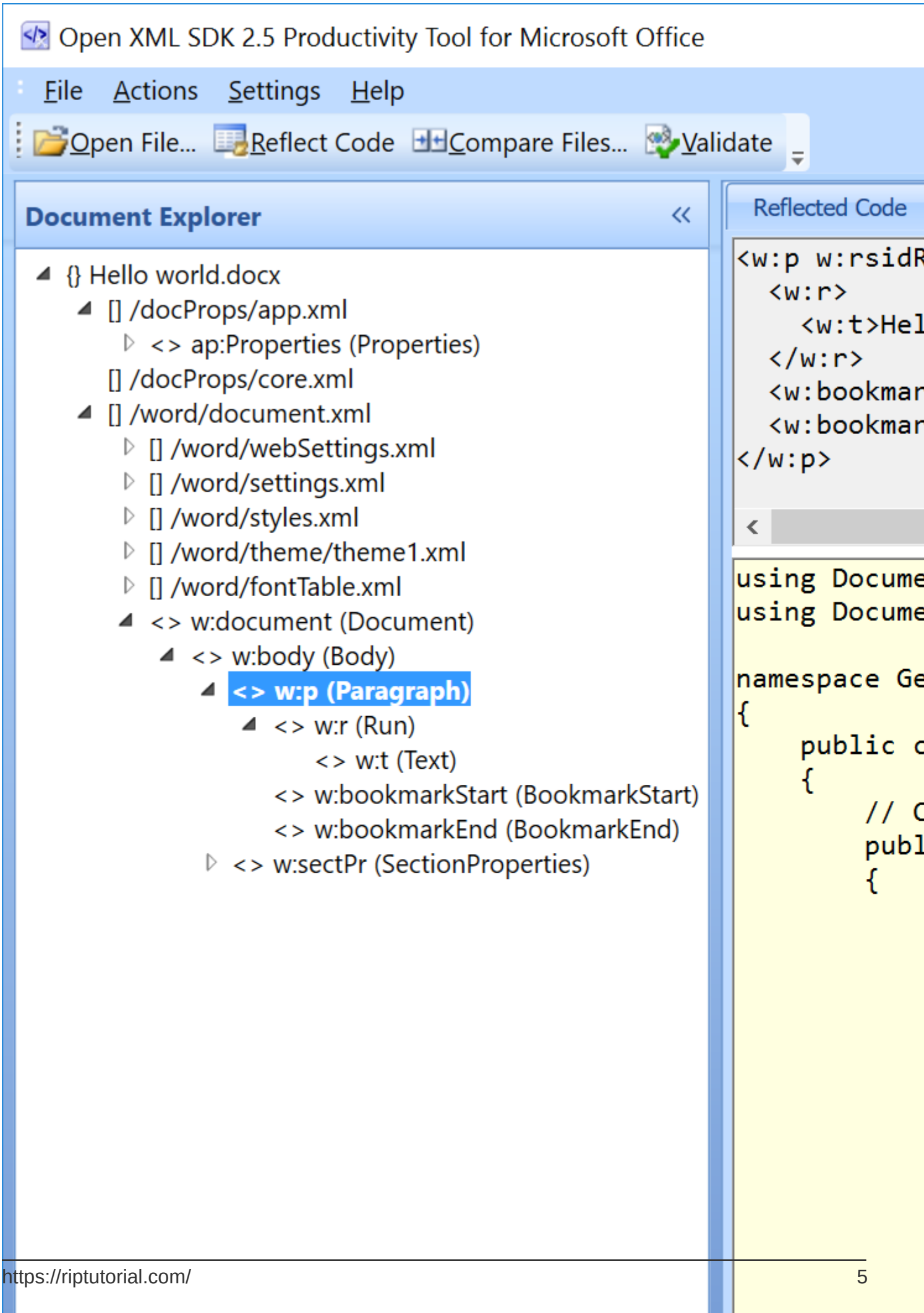


Using Open XML SDK 2.5 Productivity Tool

Reading the specification for the document formats in OpenXML can be a time consuming process. Sometimes you just want to see how to produce a certain feature in a word-document. The Open XML SDK 2.5 Productivity Tool for Microsoft Office (OpenXmlSdkTool.exe) does just that. Its main features are:

- See the structure of a file - which xml-parts does it contain
- Navigate the xml in each of these parts
- Generate c#-code for producing the selected part of the document
- Link to the file format specification describing more details
- Document OpenXML Validation

For a simple 'Hello world.docx' it looks like this:


Open XML SDK 2.5 Productivity Tool for Microsoft Office
File
Actions
Settings
Help
Open File...
Reflect Code
Compare Files...
Validate
Document Explorer
{} Hello world.docx
[] /docProps/app.xml
<> ap:Properties (Properties)
[] /docProps/core.xml
[] /word/document.xml
[] /word/webSettings.xml
[] /word/settings.xml
[] /word/styles.xml
[] /word/theme/theme1.xml
[] /word/fontTable.xml
<> w:document (Document)
<> w:body (Body)
<> w:p (Paragraph)
<> w:r (Run)
<> w:t (Text)
<> w:bookmarkStart (BookmarkStart)
<> w:bookmarkEnd (BookmarkEnd)
<> w:sectPr (SectionProperties)
Reflected Code
<w:p w:rsidR
<w:r>
<w:t>Hel
</w:r>
<w:bookmar
<w:bookmar
</w:p>
using Docume
using Docume
namespace Ge
{
public c
{
// C
publ
{

The pane on the left show the document-structure. The top-right pane displays the xml corresponding to the selection in the tree, and finally the bottom-right pane show some generated code for producing the xml displayed above it.

This enables a very hands on way to investigate a certain feature:

- Produce an example-document (fx a word-document)
- Open the document in Productivity Tool
- Use 'Reflect Code' to generate code

The SDK can be downloaded from https://www.microsoft.com/en-us/download/details.aspx?id=30425 - download and install both of the msi packages. After installation use OpenXMLSdkTool.exe installed in "C:\Program Files (x86)\Open XML SDK\V2.5\tool".

Read Getting started with openxml online: https://riptutorial.com/openxml/topic/6967/getting-started-with-openxml

Chapter 2: Create New Word Document with Open XML

Introduction

The OpenXML document markup standard is an XML based format which enables solutions on many software platforms and operating systems.

Examples

Hello World

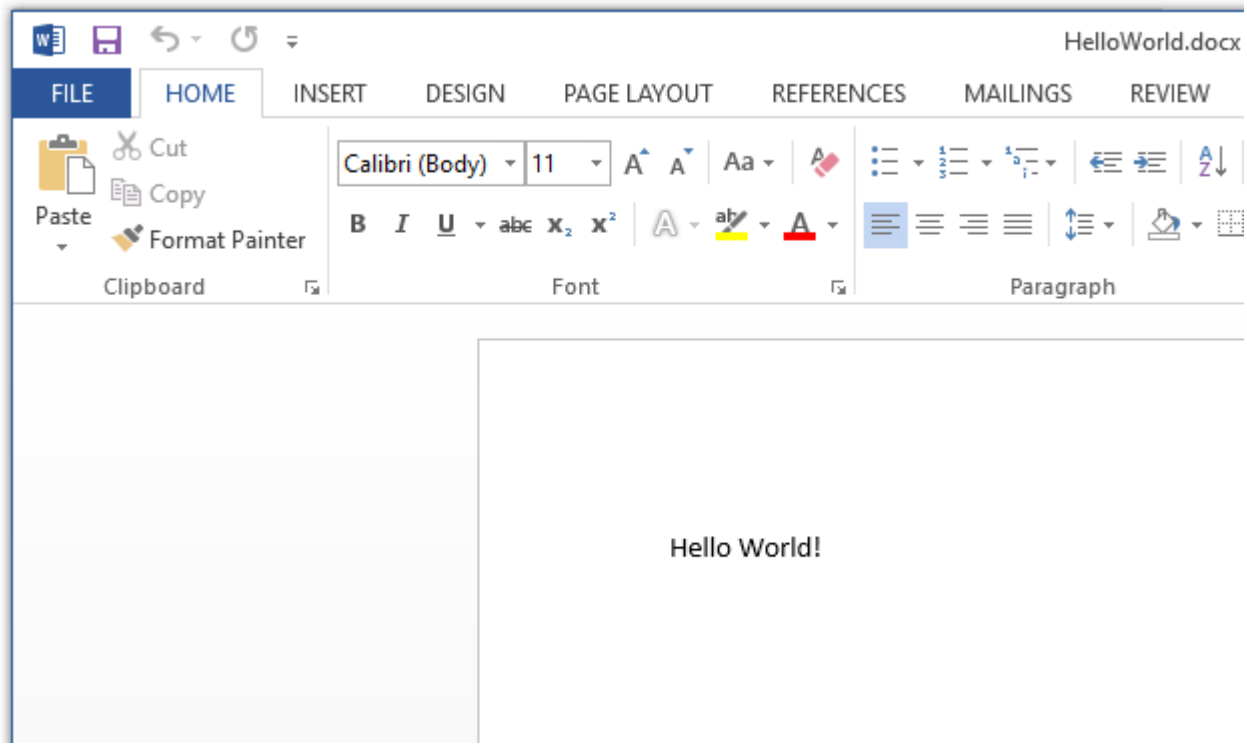
First, create a new console project using Visual Studio and add the following .dlls to your project:

```
DocumentFormat.OpenXml
WindowsBase
```

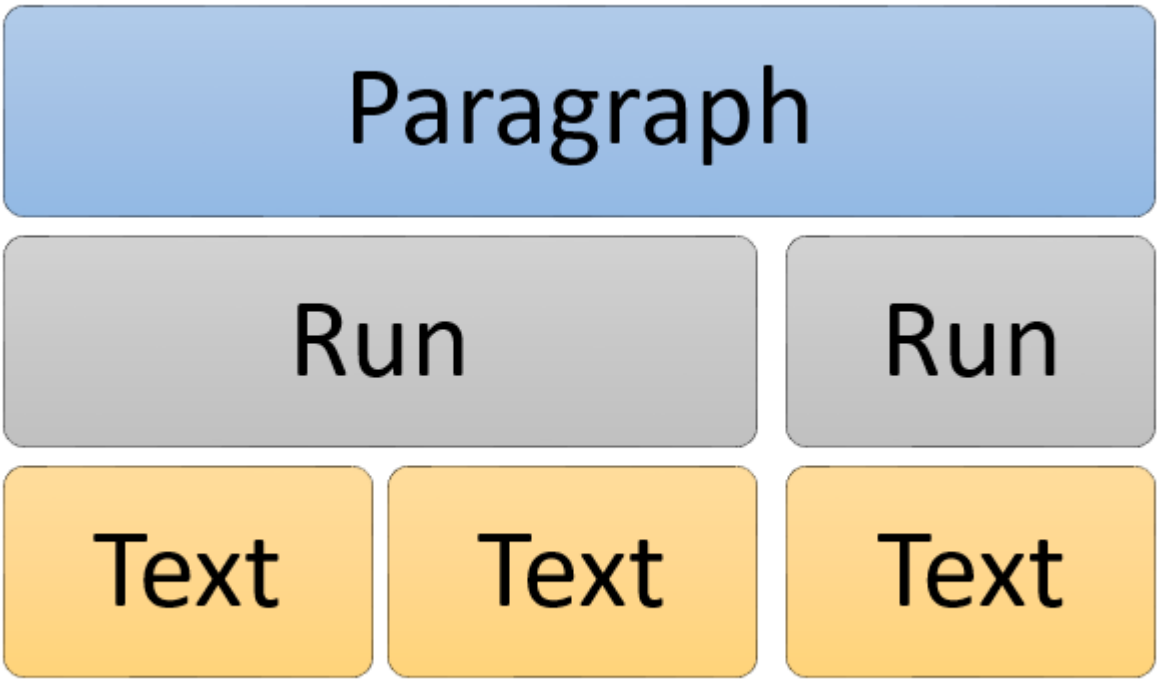
Next, compile and execute the following code:

```
static void Main(string[] args)
{
    // Create a Wordprocessing document.
    using ( WordprocessingDocument package = WordprocessingDocument.Create("HelloWorld.docx",
WordprocessingDocumentType.Document))
    {
        // Add a new main document part.
        package.AddMainDocumentPart();

        // Create the Document DOM.
        package.MainDocumentPart.Document =
            new Document(
                new Body(
                    new Paragraph(
                        new Run(
                            new Text("Hello World!")))));

        // Save changes to the main document part.
        package.MainDocumentPart.Document.Save();
    }
}
```

Under your `\bin\Debug` folder you should have your first WordprocessingML document:



The text that we added in the above example is stored under the main document part. Inside the main document part there is the **document** element which allows a child element **body** to store the text which makes our document. There are two main groups of content for the document body, block level *(paragraphs and tables)* and inline content *(runs and text)*. The block level content provides the main structure and contains inline content. To understand the example above, we first need to understand the text hierarchy in WordprocessingML. A paragraph is split into different runs. A run is the lowest level element to which formatting can be applied. The run is split up again into various text elements.



Read Create New Word Document with Open XML online:
https://riptutorial.com/openxml/topic/9833/create-new-word-document-with-open-xml

Chapter 3: How to add an image to a Word Document.

Introduction

Inserting an image in a word document using OpenXml require two actions: add the image inside the openxml and refer to the image in your Document

Remarks

if you only add the image to the openxml structure without refering it in the Word Document, the next time you "open / save" your document, the image file will be deleted.

Word delete all orphan references. So make sure to add the image in the Word Document otherwise you will need to redo all the steps.

Examples

Add the image to the OpenXml structure

```
private string AddGraph(WordprocessingDocument wpd, string filepath)
{
    ImagePart ip = wpd.MainDocumentPart.AddImagePart(ImagePartType.Jpeg);
    using (FileStream fs = new FileStream(filepath, FileMode.Open))
    {
        if (fs.Length == 0) return string.Empty;
        ip.FeedData(fs);
    }

    return wpd.MainDocumentPart.GetIdOfPart(ip);
}
```

In this case we use a FileStream to retrieve the image, but feedData(Stream) is waiting for any kind of Stream.

Refer to the image in the Word Document

```
private void InsertImage(WordprocessingDocument wpd, OpenXmlElement parent, string filepath)
{
    string relationId = AddGraph(wpd, filepath);
    if (!string.IsNullOrEmpty(relationId))
    {
        Size size = new Size(800, 600);

        Int64Value width = size.Width * 9525;
        Int64Value height = size.Height * 9525;
```

```
        var draw = new Drawing(
            new DW.Inline(
                new DW.Extent() { Cx = width, Cy = height },
                new DW.EffectExtent()
                {
                    LeftEdge = 0L,
                    TopEdge = 0L,
                    RightEdge = 0L,
                    BottomEdge = 0L
                },
                new DW.DocProperties()
                {
                    Id = (UInt32Value)1U,
                    Name = "my image name"
                },
                new DW.NonVisualGraphicFrameDrawingProperties(new A.GraphicFrameLocks() {
NoChangeAspect = true }),
                new A.Graphic(
                    new A.GraphicData(
                        new PIC.Picture(
                            new PIC.NonVisualPictureProperties(
                                new PIC.NonVisualDrawingProperties()
                                {
                                    Id = (UInt32Value)0U,
                                    Name = relationId
                                },
                                new PIC.NonVisualPictureDrawingProperties()),
                                new PIC.BlipFill(
                                    new A.Blip(
                                        new A.BlipExtensionList(
                                            new A.BlipExtension() { Uri = "{28A0092B-C50C-
407E-A947-70E740481C1C}" })
                                        )
                                    {
                                        Embed = relationId,
                                        CompressionState =
                                        A.BlipCompressionValues.Print
                                    },
                                        new A.Stretch(
                                            new A.FillRectangle())),
                                            new PIC.ShapeProperties(
                                                new A.Transform2D(
                                                    new A.Offset() { X = 0L, Y = 0L },
                                                    new A.Extents() { Cx = width, Cy = height
}),
                                                    new A.PresetGeometry(new
A.AdjustValueList()) { Preset = A.ShapeTypeValues.Rectangle }))) { Uri =
"http://schemas.openxmlformats.org/drawingml/2006/picture" })
                                                    )
            {
                DistanceFromTop = (UInt32Value)0U,
                DistanceFromBottom = (UInt32Value)0U,
                DistanceFromLeft = (UInt32Value)0U,
                DistanceFromRight = (UInt32Value)0U,
                EditId = "50D07946"
            });

        parent.Append(draw);
    }
}
```

In this example I set a static size of 800*600 but you can set any size you need

Read How to add an image to a Word Document. online: https://riptutorial.com/openxml/topic/9397/how-to-add-an-image-to-a-word-document-

Chapter 4: Insert image into an inline "inline shape" in word documents

Introduction

Insert an image into an MS Word document shapes such as Rectangles and ovals.

This documentation assumes you know how to insert an image into a word document, open and close a word document using OpenXML

Examples

Add the following OpenXML namespaces to your class

```
using System;
using System.Collections.Generic;
using System.Linq;
using DocumentFormat.OpenXml;
using A = DocumentFormat.OpenXml.Drawing;
using DW = DocumentFormat.OpenXml.Drawing.Wordprocessing;
using PIC = DocumentFormat.OpenXml.Drawing.Pictures;
using DocumentFormat.OpenXml.Drawing.Wordprocessing;
using Wps = DocumentFormat.OpenXml.Office2010.Word.DrawingShape;
```

Open the document and add imagePart object to reference the picture you want to insert into the shape

Now open the document using OpenXML, you must add an imagePart that references the picture object to the MainDocumentPart object by using a file stream, and get the ID of the image

```
    string temp;
MainDocumentPart mainPart = document.MainDocumentPart;
                                ImagePart imagePart = mainPart.AddImagePart(ImagePartType.Bmp);

                                using (FileStream stream = new FileStream(barcodepath,
FileMode.Open))
                                {
                                   imagePart.FeedData(stream);
                                }

                                temp = mainPart.GetIdOfPart(imagePart);
```

Get a reference of a Blip object

In office OpenXML, a picture that is inserted into a word document is considered a "Blip" Object or element. The class is derived from the DocumentFormat.OpenXml.Drawing the Blip must have an Embed value that is an imagePart ID. The Blip object then goes inside a BlipFill Object/element,

and that also goes inside a graphicData Object/element and that in turn goes into a graphic object element. I'm pretty sure by now you've realized everything works like an XML tree. Sample Open XML tree below.

```
<a:graphic xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main">
                      <a:graphicData
uri="http://schemas.microsoft.com/office/word/2010/wordprocessingShape">
                        <wps:wsp>
                          <wps:cNvSpPr>
                            <a:spLocks noChangeArrowheads="1" />
                          </wps:cNvSpPr>
                          <wps:spPr bwMode="auto">
                            <a:xfrm>
                              <a:off x="0" y="0" />
                              <a:ext cx="1234440" cy="1234440" />
                            </a:xfrm>
                            <a:prstGeom prst="roundRect">
                              <a:avLst>
                                <a:gd name="adj" fmla="val 16667" />
                              </a:avLst>
                            </a:prstGeom>
                            <a:blipFill dpi="0" rotWithShape="1">
                              <a:blip r:embed="Raade88ffea8d4c1b" />
                              <a:stretch>
                                <a:fillRect l="10000" t="10000" r="10000" b="10000" />
                              </a:stretch>
                            </a:blipFill>
                          </wps:spPr>
                          <wps:bodyPr rot="0" vert="horz" wrap="square" lIns="91440"
tIns="45720" rIns="91440" bIns="45720" anchor="t" anchorCtr="0" upright="1">
                            <a:noAutofit />
                          </wps:bodyPr>
                        </wps:wsp>
                      </a:graphicData>
                    </a:graphic>
```

Adding reference of the image the shapes in the template document.

Now you have a reference of the image. Insert the image into the shapes in the template document. To do this, you will have to use some LINQ to iterate through the document and get a reference to all the shapes in the document. The wps:spPr element you see in the above XML code is the xml element for the shapes in a document. The equivalent C# class is WordprocessingShape

```
IEnumerable<DocumentFormat.OpenXml.Office2010.Word.DrawingShape.WordprocessingShape> shapes2 =
document.MainDocumentPart.document.Body.Descendants<DocumentFormat.OpenXml.Office2010.Word.DrawingShape
```

With a collection reference of all the shapes, loop through the collection.

Now that you have a collection of all the shape references in the document. Loop through the collection with a foreach, and through each iteration create a Blip object. Set the Blip object embed value to the picture ID reference you captured earlier from the image part. Also create a Stretch object, and FillRectangle object(these are not really necessary, they are just used them for proper

alignment of the image). And append each to its parent object like the XML tree equivalent.

```
foreach (DocumentFormat.OpenXml.Office2010.Word.DrawingShape.WordprocessingShape sp in
shapes2)
                                {
                                    //Wps.ShapeProperties shapeProperties1 =
                                    A.BlipFill blipFill1 = new A.BlipFill() { Dpi =
(UInt32Value)0U, RotateWithShape = true };
                                    A.Blip blip1 = new A.Blip() { Embed = temp };

                                    A.Stretch stretch1 = new A.Stretch();
                                    A.FillRectangle fillRectangle1 = new A.FillRectangle() {
Left = 10000, Top = 10000, Right = 10000, Bottom = 10000 };
                                    Wps.WordprocessingShape wordprocessingShape1 = new
Wps.WordprocessingShape();



                                        stretch1.Append(fillRectangle1);
                                        blipFill1.Append(blip1);
                                        blipFill1.Append(stretch1);
                                        Wps.ShapeProperties shapeProperties1 =
sp.Descendants<Wps.ShapeProperties>().First();
                                        shapeProperties1.Append(blipFill1);

                                }
```

Read Insert image into an inline "inline shape" in word documents online: https://riptutorial.com/openxml/topic/9660/insert-image-into-an-inline--inline--shape--in-word-documents

Credits

S. No	Chapters	Contributors
1	Getting started with openxml	Community, Daniel Brixen, Mandar Badve, Taterhead
2	Create New Word Document with Open XML	FortyTwo
3	How to add an image to a Word Document.	Maxime Porté
4	Insert image into an inline "inline shape" in word documents	Dankwansere