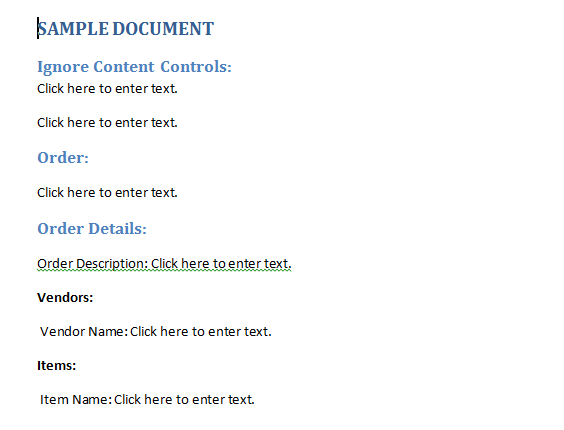
# Overview:

This utility populates the Word templates with data and generates the final document. It has been created in Visual Studio 2010 and uses Open Xml 2.0 SDK which can be downloaded from <http://www.microsoft.com/download/en/details.aspx?id=5124>.

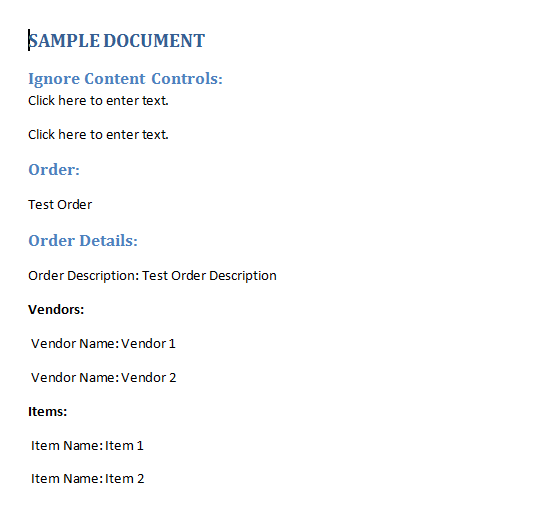
The purpose of creating this utility was to use the Open Xml 2.0 SDK to generate Word documents based on predefined templates using minimum code changes. These documents can either be refreshable or non- refreshable. I’ll explain this difference later.

A few samples for generating Word 2010 documents have been provided. The samples that generate reports using data bound content controls have been added. More samples can be added later as per feedback. The screenshots below display the sample template and the document generated out of this template using this utility.

**Sample Template:**

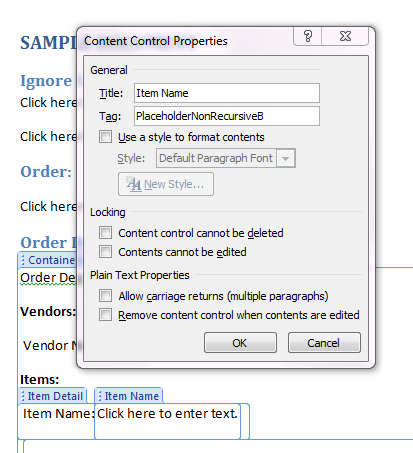


# Document Generated from Sample Template:



# Template Design:

The sample templates are inside “WordDocumentGenerator.Client\Sample Templates” folder. A content control as displayed below can have Title and Tag properties.



The logic is to have a tag property of a content control and then populate data using that placeholder. This means every content control inside a Word template will have a different Tag.

As per image above the tag of the content control is “PlaceholderNonRecursiveB”. During document generation we can assign (not mandatory) a unique Id e.g. Guid of a record to make the tag unique e.g. “PlaceholderNonRecursiveB:Guid”. Let’s say that we have an Id and Name field. Thus the Name will be the content of the control and tag will be “PlaceholderNonRecursiveB:Id”. As per Word 2010 the Tag maximum length is 64.

In code we map the tag to the PlaceHolderType enum.

public enum PlaceHolderType

    {

        None = 0,

        Recursive = 1,

        NonRecursive = 2,

        Ignore = 3,

        Container = 4

    }

There can be multiple types of PlaceHolders

* Recursive: This type corresponds to controls where there is 1:N relation between template and data i.e. one example will be repeating a list of Items.
* Non-Recursive: This type corresponds to controls where there is 1:1 relation between template and data i.e. one example will be showing a User name.
* Ignore: No action is required for these controls.
* Container: This type is required only for refreshable documents. We save the container region in CustomXmlPart the first time document is generated from template. Next time onwards we retrieve the container region that was saved and refresh the document. This makes the document self-refreshable.

# I’ve named the tags in “Test\_Template – 1.docx” as per their PlaceHolder type to make it more clear.

# Code Implementation:

As explained above the Tag property will used to bind data to this content control. The project “WordDocumentGenerator.Library” is the utility library. The project “WordDocumentGenerator.Client” shows how this library can be used to generate documents.

“DocumentGenerator” is the base class that needs to be inherited by the document generators. The sample generators location is “WordDocumentGenerator.Client\Sample Document Generators”.

~~I’ve not used DataBinding as displayed below as rich text content controls don’t support data binding. Read more about DataBinding at~~ [~~http://msdn.microsoft.com/en-us/library/bb398244.aspx~~](http://msdn.microsoft.com/en-us/library/bb398244.aspx)

~~<w:dataBinding w:prefixMappings="xmlns:ns0='http://contoso.com/2005/contracts/commercialSale'~~

~~" w:xpath="/ns0:contract[1]/ns0:dateExecuted[1]" w:storeItemID="{ABB284D9-2C5E-41BD-A2F2-B5FC934955A9}"/>~~

~~I’ll try to incorporate DataBinding to plain text controls in the Utility later if required.~~

Samples for Data bound content controls have been added. The utility has been updated.

In order to protect the document I’ve used already created salt and hash. For an implementation where one needs to have document protection enabled for custom passwords one can view <http://blogs.msdn.com/b/vsod/archive/2010/04/05/how-to-set-the-editing-restrictions-in-word-using-open-xml-sdk-2-0.aspx>

# Summary

The purpose of creating this utility was to use the Open Xml 2.0 SDK to generate Word documents based on predefined templates using minimum code changes. These documents can either be refreshable or non- refreshable. This utility is kind of POC/Samples rather than a full-fledged framework.