|  |  |  |
| --- | --- | --- |
| J2DOCX  point N write | Application Programming  Interface(API)  Reference Documentation.  -Vinothkumar.P.T., | |
| J2DOCX-“Java to docx” is a Java API to write and manipulate the contents over the Microsoft docx document programmatically. This API comprises set of useful functions and features to work with docx. The version 2.1 supports writing of Text, Image, Hyperlink and Custom xml structures. | | Release 2.1 |

Copyright © 2012-2013

Copies of this document may be made for your own use and for distribution to others, provided that you do not charge any fee

for such copies and further provided that each copy contains this Copyright Notice, whether distributed in print or electronically.

Table of Contents

1. [**Understanding Microsoft Word Document Docx**](#understand_docx)**3**

**2.** [**Exploring docx**](#explore_docx)**4**

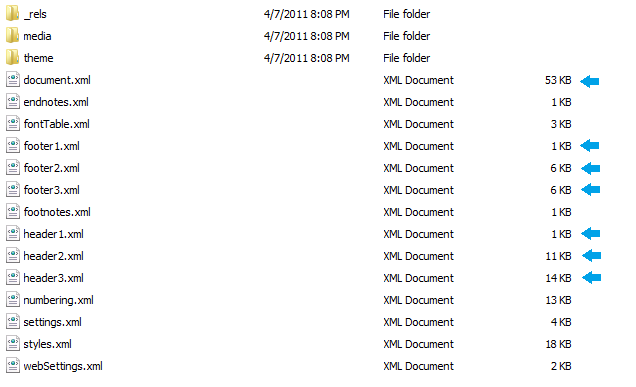
1. [**Extracting and Archiving (Unzipping and Zipping) the docx file**](#ext_arcve_docx)**4**
2. [**Bookmark and Content Controls**](#bm_n_cc)**10**
3. [**The J2DOCX API**](#j2docx_api)**13**
4. [**Quick Facts**](#quick_facts)**13**
5. [**Specifications**](#specifications)**13**
6. [**Package Details**](#package_details)**14**
7. [**Sample Programs**](#sample_programs)**16**

**1. Understanding Microsoft Word Document Docx**

The Microsoft’s “.docx” word document is the replacement of the earlier version“.doc”.

.doc document is made up of a single xml called “WordML” in which the header, body and footer contents resides whereas in .docx, all these are clustered and a separate xml file is provided for each a section like header.xml, footer.xml, document.xml(body part of the document) and relationship files to couple altogether makes the doc+xml= docx

**1.1 The Docx Skeleton:**



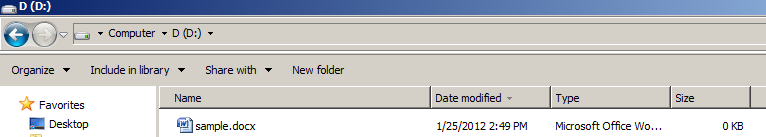
The above skeleton shows the document.xml, footer\*.xml and header\*.xml, in this case the document might be having 3 pages so three header’s and three footer’s xml could be seen, but this is not necessary to be having footer and header counts based on the page number, this could be the contents are being shared among these files. header\*.xml and footer\*.xml will not be present in case of no header and footer are created by the user.

But there could be only one document.xml per docx document irrespective of the page count and document size.

**Exploring docx:**

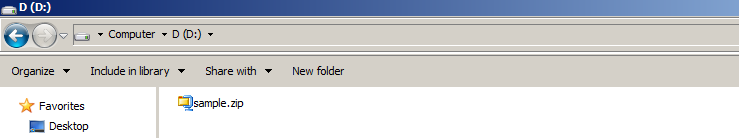
**Extracting and Archiving (Unzipping and Zipping)the docx file:**

Open your Microsoft office word 2007 or 2010 and save it on disk, ensure the file format is “.docx”. In this guide, the file “sample.docx” is used.

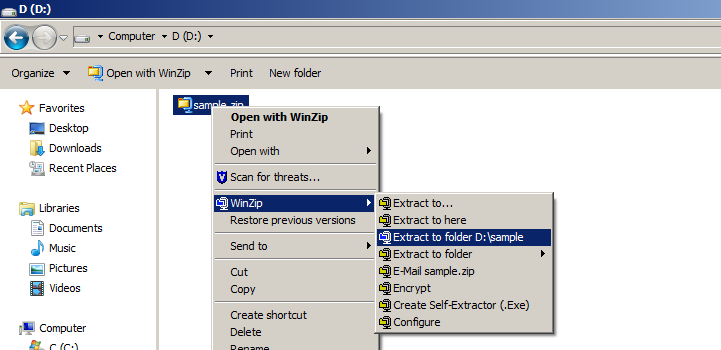


**To Extract:**

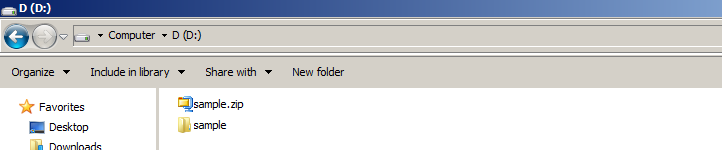
Rename the file from “.docx” to “.zip”. Please refer the screen shot.



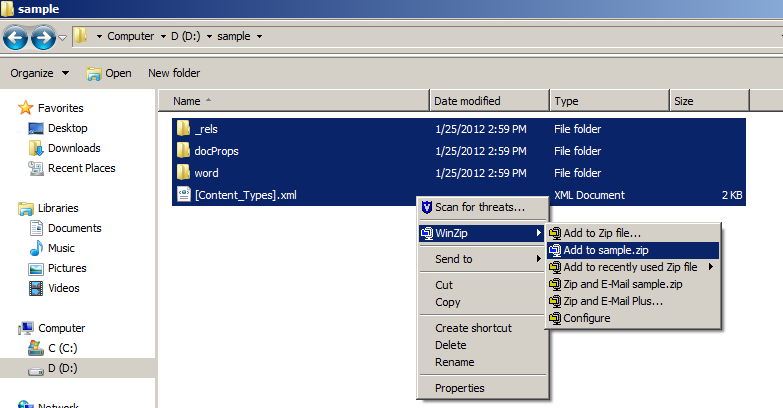
Extract it using your zip utility that is installed in your machine.



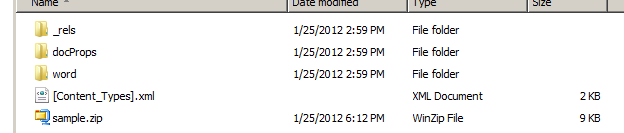
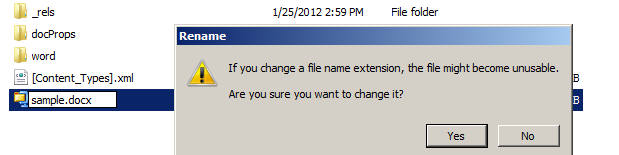
Here extracted to a directory “sample” same as that of the file name.



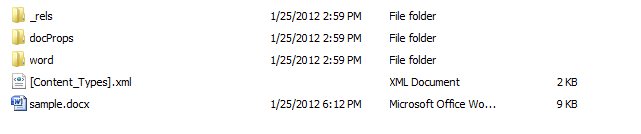
**To Archive:**

Go to the root directory “/sample”, and select all inside of it then add to the zip. 

Now your zip file is ready.

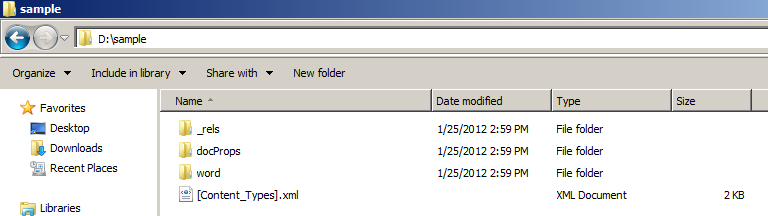
****Rename it as “.docx” back.****

Click ‘Yes’, now your docx file has been successfully created back.

****

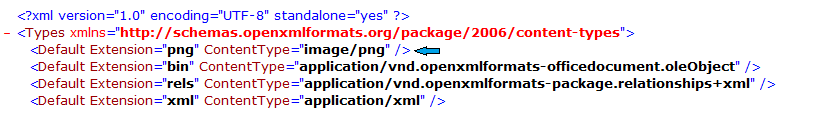
**The root directory:**

The root directory “/sample”hasthe directories \_rels, docProps, word and a XML file called [Content\_Types].xml as there in the screen shot.



Here the directory “/word” and “[Content\_Types].xml” is our objective for programming.

**[Content\_Types].xml**- describes media types, font, style definitions etc used across the document.

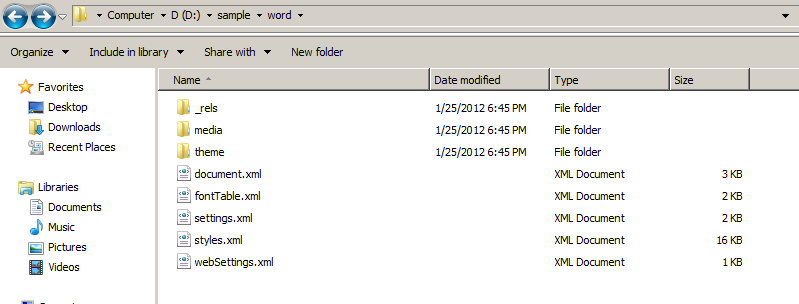
E.g. if we want to use “png” image file in the document (only when inserting programmatically), should have to add an entry as below.If it is missed, word document will be failed to open.

When adding images Non-programmatically, word constructs this element automatically.

**/word:**

Directory is the core of the document because it has all the contents necessary in display perspective. The contents of the “/word” directory is shown already in the skeleton section. Let’s see elaborately.

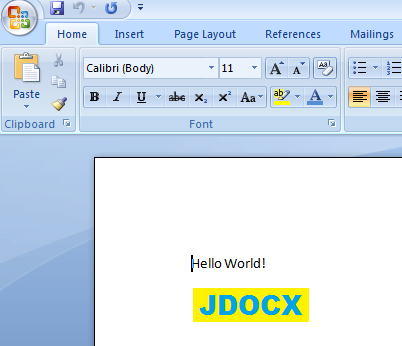
The basic components inside of it are



* **/\_rels** - Contains relationship configuration file.
* **/media**–contains inserted media files such as images, video clips… etc. And this directory is created only after inserting at least one media file into the document.
* **document.xml**-describes the body of the document
* **header\*.xml & footer\*.xml-** optional if header and footer created in the document.

Let us see how the document is being orchestrated.

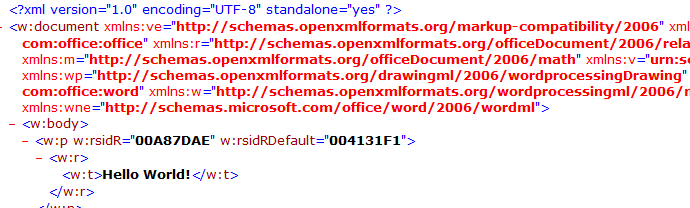
Type some text and paste an image on the “sample.docx” as below.



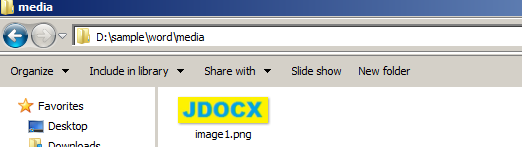
Extract the file as we saw in the section Explore docx.

**Text:**

Open the document.xml, search for the text “hello world”, will be present as below.

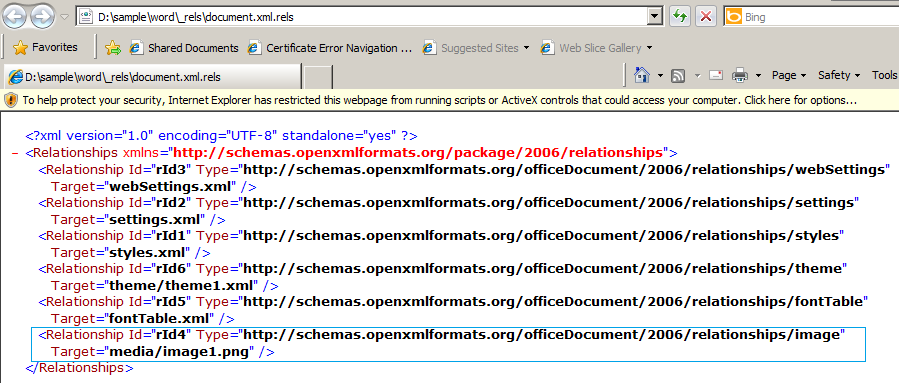


**Image:**

The image will be present in the directory /media as below.  


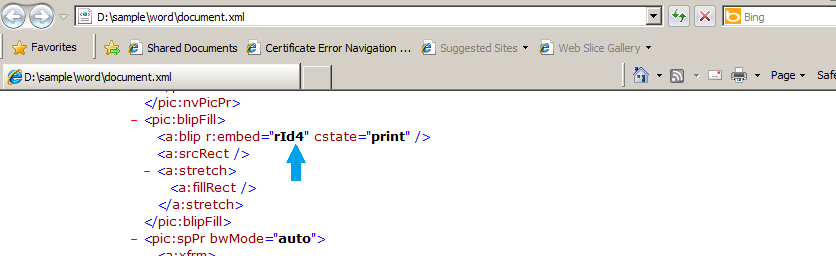
The relationship between the media file present in the /media directory and the body of the document is built up by the “document.xml.rels” present in the “/\_rels” directory as below.

**Showing: word/\_rels/document.xml.rels:**



Here the Relationship Id “rId4” is generatedand the path specified in the “Target” attribute, the same Id will be used to make the coupling with the “document.xml”. This Id should be unique.

Please find the “rId4” in the document.xml as below.



Thus the image appeared on the document’s body. The below paradigm explains the relationship among the media file and document.xml

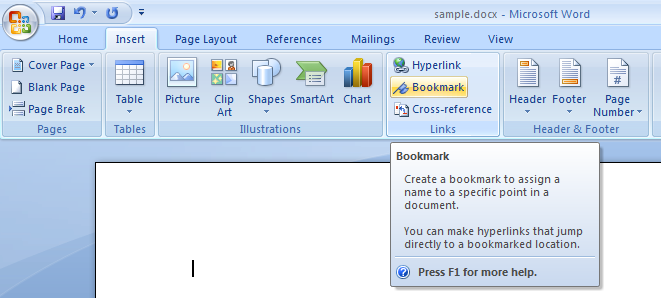
**Bookmark and Content Controls:**

**The bookmark:**

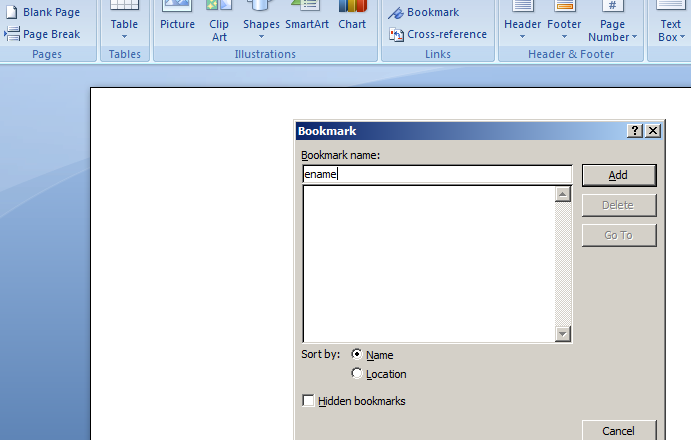
Bookmark is a simple place holder used to denote a particular location on a document used to position the contents like text, images.

**Creating bookmark:**

Place the cursor on the document where needs to create a bookmark, and then navigate to the Insert tab, click bookmark.



Give a name to the bookmark



The same name will be used to refer to that particular bookmark.

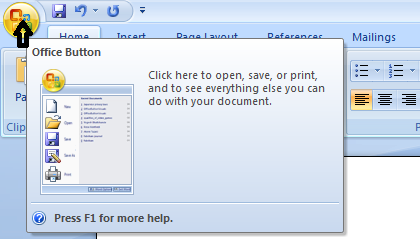
**Content Controls:**

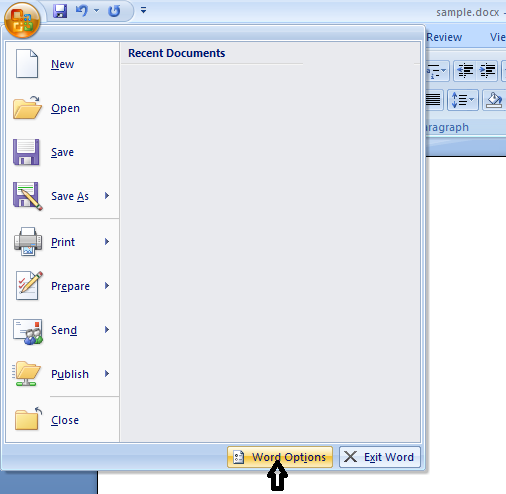
Content control is also like a place holder but it has advance features,

* + It is used to hold custom xml and rich contents like tables, grids and etc...
  + We can protect contents from being edited by the other users so read only contents can be created on the document.
  + There are multiple content control type’s are available (See Microsoft’s Specification).

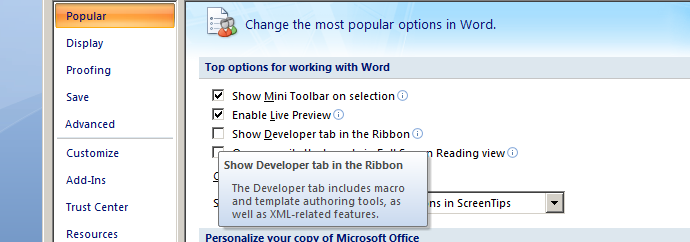
**Creating Content Control:**

Content controls are available in the developer tab on the MS Word. If the developer tab is not enabled, click on the ribbon icon and then Word Options.

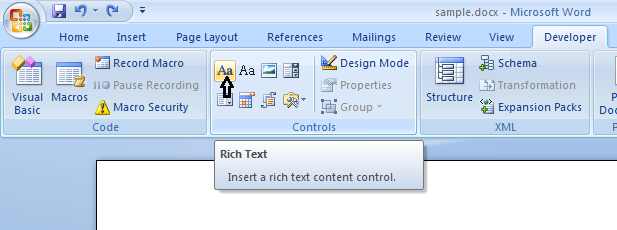




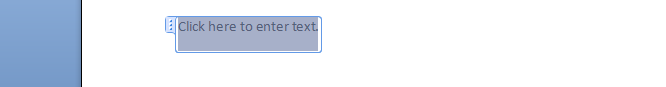
Enable the Check box “Show Developer tab in the Ribbon”.



Click the Developer tab, Create Rich text content control as below.



The created Content Control would be looking like this,



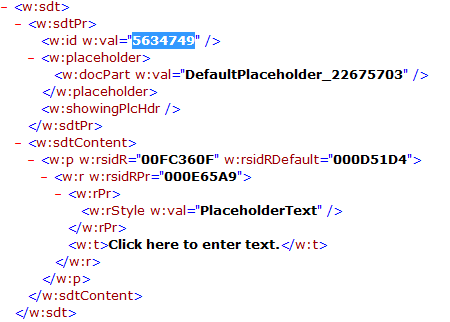
**Retrieve the Content control id:**

Content control ID is a unique and used to locate a particular content control, it is a random number generated by MS Word itself. To retrieve the id, Please follow the below steps,

Extract document (Please refer the section: Extracting and Archiving (Unzipping and Zipping) the docx file) or Open the document with any zip utility,

Open the corresponding xml file like document.xml/header\*.xml,.

The id value would be available under the w:sdt tag as highlighted below. J2docx is used this id to write custom xml. This id is editable; developer can give any number as id but it must be unique per document.

****

We have learnt how to create the bookmarks and content controls, let’s see how J2DOCX is used to write contents programmatically.

**THE J2DOCX API:**

The J2DOCX 2.1 is a java approach to write the contents programmatically over thedocx documents. It has set of useful functions to manipulate docx.

**QUICK FACTS:**

* The first open source API to write microsoft word 2007/2010 documents.
* It provides template-programming model. Userscan simply design the template and fill it up with the contents by j2docx API using "bookmarks" and ”content control” concepts. Remember that it will not create new word document, it writes existing document templates.
* Bookmark is the feature in microsoft word by which we can create place holders to point a particular location in the word document, anywhere in header, body and footer.
* j2docx's underlying mechanism is "Point and Write", which means that it writes over the bookmark that already pointed by the user on the document template.
* We can’t leave vertical and horizontal spaces programmatically. It is default word’s behavior. It adjusts automatically based on the content size.
* User can format the style of the contents either programmatically or Non programmatically.
  + In programmatically way, need to pass parameters like font-family, font-size, font-color etc.
  + In “Non Programmatically” way, On selecting the bookmark set the formatting style in the word font dialog box, then leave a space or type any dummy character/word. This will be ensured by j2dox to preserve the style that already set by the user to the corresponding bookmark. In this case, J2docx will simply replace and write the contentsbut not the font styles. This is the beauty of j2docx, because it minimizes the programming effort.

i-e: (**create bookmark + format it + leave a space + supply bookmark name to j2docx**) .

* J2docx accepts only below image formats to write over the document.

JPEG/JPG/PNG/GIF

**SPECIFICATIONS:**

**Library Dependencies:**

J2DOCX Version 1.1 was built up with the following libraries, so these should be included.

|  |  |  |
| --- | --- | --- |
| NAME | VERSION | URL |
| JDOM | 1.x (or) 1.1.3\* | http://www.jdom.org/downloads/ |
| JAXEN | 1.x (or) 1.1.6\* | http://jaxen.codehaus.org/releases.html |

**Package Details:**

|  |  |
| --- | --- |
| **com.j2docx.operations** | This package contains classes that perform common operations over the document like extract and archive the document. |
| **com.j2docx.spray** | Contains classes used to write the contents. |
| **com.j2docx.util** | Contains the helper classes. |
| **com.j2docx** | Contains the exception classes. |

**Package: com.j2docx.operations**

|  |
| --- |
| **Class Details:** DOCXHandler.java |
| **Method Details:** |
| |  | | --- | | **public void doExtract(String sourceDocx, String destinationDirectory)**   * **It extracts source docx file to the given destination directory.** | | sourceDocx - is the absolute path of the docx file. | | destinationDirectory – is the destination where the file has to be extracted. |   It throws,   * DocxHandlerException- if any issue in Extracting/Archiving docx file. * FileNotFoundException- if source is not available in the path. * IO Exception- if any issue in reading/writing files. |
| |  | | --- | | **public void doZip(String sourceDir,String targetDocxPath)**   * **It packs the files in the source directory and makes docx file.** | | sourceDir- path of the directory where all the files are wrapped in. | | targetDocxPath - destination path of the docx to be created. |   It throws,   * DocxHandlerException- if not found a valid root directory contains docx contents. * FileNotFoundException- if source is not available in the path. * IOException- if any issue in reading/writing files.  |  | | --- | | **public ArrayList getFilesToSpray (String directoryPath)**   * **Returns ArrayList containing all the necessary configuration files (xml) path of the docx.** | | directoryPath –the extracted file path. | |

**Package: com.j2docx.spray**

|  |
| --- |
| **Class Details:** BookmarkSprayer.java |
| **Method Details:** |
| |  | | --- | | **public void sprayText(String extractedPath, String bookmarkName, String bookmarkValue,**  **String style)**   * **It writes text on the document at the specified bookmark, font style can be customized by specifying style properties, Otherwise default font is Arial 18pt.** | | extractedPath - Path of the root directory of the extracted docx document. | | bookmarkName - User defined bookmark name. | | bookmarkValue - The value to be written over the document at the specified bookmark. | | style - Defines style of the font as attributes separated by comma. It can be null or empty string to specify default value.  Example:  **font-family:verdana;font-weight:bold;font-size:11;font-color:red;italic:true;uline:true**  Here,  font-family: Supported font family names like Arial, Verdana., etc.  font-weight: Specifies Bold letters.  font-size: Specifies the size of the font.  font-color: Specifies color of the font.  italic: Specifies font is needed in italic .  uline: Specifies underlined letters. |   It throws,   * DocxHandlerException- if any issue writing docx file. * FileNotFoundException- if source is not available in the path. * IO Exception- if any issue in reading/writing files. * JDOMException- If any exception in parsing xml elements.  |  | | --- | | **public void sprayTextTargetFile (String xmlFilePath, String bookmarkName, String bookmarkValue, String style)**   * **It writes text on the document at the specified bookmark, font style can be customized by specifying style properties, Otherwise default font is Arial 18pt.** | | xmlFilePath - Path of the XML document to be written. | | bookmarkName - User defined bookmark name. | | bookmarkValue - The value to be written over the document at the specified bookmark. | | style- Defines style of the font as attributes separated by comma. It can be null or empty string to specify default value.  Example:  **font-family:verdana;font-weight:bold;font-size:11;font-color:red;italic:true;uline:true**  Here,  font-family: Supported font family names like Arial, Verdana., etc.  font-weight: Specifies Bold letters.  font-size: Specifies the size of the font.  font-color: Specifies color of the font.  italic: Specifies font is needed in italic .  uline: Specifies underlined letters. |   It throws,   * DocxHandlerException- if any issue writing docx file. * FileNotFoundException- if source is not available in the path. * IO Exception- if any issue in reading/writing files. * JDOMException- If any exception in parsing xml elements. |
| |  | | --- | | **public void sprayImage(String docExtractedPath, String bookmarkName, int height, int width,**  **String imageAbsolutePath)**   * **It writes image on the document at the specified bookmark.** * **It supports only the following image formats**   **\*.jpg,\*.jpeg,\*.png,\*.gif** | | docExtractedPath- Path of the root directory of the extracted docx document. | | bookmarkName- User defined bookmark name. | | Height- Specifies image height. | | Width- Specifies image width. | | imageAbsolutePath- Path of the source image file to be written over the document. |   It throws,   * DocxHandlerException- if any issue writing docx file. * FileNotFoundException- if source is not available in the path. * IOException- if any issue in reading/writing files. * JDOMException- If any exception in parsing xml elements.  |  | | --- | | **public void sprayImageTargetFile (String docExtractedPath, String xmlFilePath,**  **String bookmarkName, int height, int width, String imageAbsolutePath)**   * **It writes image on the document at the specified bookmark.** * **It supports only the following image formats**   **\*.jpg,\*.jpeg,\*.png,\*.gif** | | docExtractedPath- Path of the root directory of the extracted docx document. | | xmlFilePath - Path of the XML document to be written. | | bookmarkName- User defined bookmark name. | | Height- Specifies image height. | | Width- Specifies image width. | | imageAbsolutePath- Path of the source image file to be written over the document. |   It throws,   * DocxHandlerException- if any issue writing docx file. * FileNotFoundException- if source is not available in the path. * IOException- if any issue in reading/writing files. * JDOMException – If any exception in parsing xml elements.  |  | | --- | | **public void sprayHyperLink (String docExtractedPath, String bookmarkName, String targetName, String targetURLValue)**   * **It writes image on the document at the specified bookmark.** * **It supports only the following image formats**   **\*.jpg,\*.jpeg,\*.png,\*.gif** | | docExtractedPath- Path of the root directory of the extracted docx document. | | bookmarkName- User defined bookmark name. | | targetName – The value to be displayed on the document for the Hyperlink | | targetURLValue - Specifies the target URL value |   It throws,   * DocxHandlerException- if any issue while writing docx file. * FileNotFoundException- if source is not available in the path. * IOException- if any issue whilereading/writing files. * JDOMException – If any exception while parsing xml elements.  |  | | --- | | **public void sprayHyperLinkTargetFile (String docExtractedPath, String xmlFilePath,**  **String bookmarkName, String targetName, String targetURLValue)**   * **It writes image on the document at the specified bookmark.** * **It supports only the following image formats**   **\*.jpg,\*.jpeg,\*.png,\*.gif** | | docExtractedPath- Path of the root directory of the extracted docx document. | | xmlFilePath - Path of the XML document to be written. | | bookmarkName- User defined bookmark name. | | targetName – The value to be displayed on the document for the Hyperlink | | targetURLValue - Specifies the target URL value |   It throws,   * DocxHandlerException- if any issue while writing docx file. * FileNotFoundException- if source is not available in the path. * IOException- if any issue whilereading/writing files. * JDOMException – If any exception while parsing xml elements.  |  | | --- | | **public void sprayCustomXML (String extractedPath, String contentControlID,**  **String inputXML)**   * **It writes custom xml contents at the specified content control id.** | | docExtractedPath-Path of the root directory of the extracted docx document. | | contentControlID-Auto generated content control id from the file. | | inputXML –Input custom xml. |   It throws,   * DocxHandlerException- if any issue when writing the docx file. * FileNotFoundException- if source is not available in the path. * IOException- if any issue whilereading/writing files. * JDOMException – If any exception while parsing xml elements.  |  | | --- | | **public void sprayCustomXMLTargetFile (String xmlFilePath, String contentControlID,**  **String inputXML)**   * **It writes custom xml contents at the specified content control id.** | | xmlFilePath - Path of the XML document to be written. | | ContentControlID-Auto generated content control id from the file. | | inputXML –Input custom xml. |   It throws,   * DocxHandlerException- if any issue when writing the docx file. * FileNotFoundException- if source is not available in the path. * IOException- if any issue while reading/writing files. * JDOMException – If any exception while parsing xml elements. |

**Package: com.j2docx.util**

|  |
| --- |
| **Class Details:** DOCXUtil.java |
| **Method Details:** |
| |  | | --- | | **public void sprayText(String extractedPath, String bookmarkName, String bookmarkValue,**  **String style)**   * **It writes text on the document at the specified bookmark, font style can be customized by specifying style properties, Otherwise default font is Arial 18pt.** | | extractedPath - Path of the root directory of the extracted docx document. | | bookmarkName - User defined bookmark name. | | bookmarkValue - The value to be written over the document at the specified bookmark. | | style - Defines style of the font as attributes separated by comma. It can be null or empty string to specify default value.  Example:  **font-family:verdana;font-weight:bold;font-size:11;font-color:red;italic:true;uline:true**  Here,  font-family: Supported font family names like Arial, Verdana., etc.  font-weight: Specifies Bold letters.  font-size: Specifies the size of the font.  font-color: Specifies color of the font.  italic: Specifies font is needed in italic .  uline: Specifies underlined letters. |   It throws,   * DocxHandlerException- if any issue writing docx file. * FileNotFoundException- if source is not available in the path. * IO Exception- if any issue in reading/writing files. * JDOMException- If any exception in parsing xml elements. |

**Sample Programs:**

Sample programs were moved under the directory “j2docx-2.1.RELEASE\docs\demo” in the release package.