Article
Eduard Lebedyuk · Mar 1, 2017 17m read

Parsing docx using XSLT

The task of handling office documents, namely docx documents, xlsx tables and pptx presentations is quite complicated. This article offers a way to parse, create and edit documents using only XSLT and ZIP.

Why? docx is the most popular document format, so the ability to generate and parse this format would always can be useful. The solution in a form of a ready-made library, can be problematic for several reasons:

- library may not exist
- you do not need another black box in your project
- library restrictions : platforms, features, etc.
- licensing
- processing speed

In this article, I would use only basic tools for work with the docx documents.

Docx structure

What is a docx document? A docx file is a zip archive which physically contains 2 types of files:

- xml files with xml or rels extensions
- media files (images, etc.)

And logically of 3 types of elements:

- Content Types - a list of media types (e.g. png) used in the document and document parts (e.g. document, page header).
- Parts - separate document parts. For docx document it is document.xml, including xml documents and media files.
- Relationships identify document parts for linkage (e.g. connection between document section and page header), external parts are also defined here (e.g. hyperlinks).

It is described in detail in the ECMA-376: Office Open XML File Formats, the main part of it is the PDF document consisting of 5,000 pages, there are also 2,000 pages of bonus content.

Minimal docx

The simplest docx after unpacking looks like this:

Let's take a look inside.

[Content_Types].xml

This file is located in the document root and lists all MIME types present in the document:

```
<Types xmlns="http://schemas.openxmlformats.org/package/2006/content-types">
    <Default Extension="rels" ContentType="application/vnd.openxmlformats-
package.relationships+xml"/>
    <Default Extension="xml" ContentType="application/xml"/>
    <Override PartName="/word/document.xml"
              ContentType="application/vnd.openxmlformats-
officedocument.wordprocessingml.document.main+xml"/>
</Types>
```

_rels/.rels

The main list of document parts. In this case, only one defined link exists - matching rId1 identifier and word/document.xml file - the main body of the document.

```
<Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
    <Relationship
        Id="rId1"
        Type="
http://schemas.openxmlformats.org/officeDocument/2006/relationships/officeDocument"
        Target="word/document.xml"/>
</Relationships>
```

word/document.xml

Main document content.

```
<w:document xmlns:w="http://schemas.openxmlformats.org/wordprocessingml/2006/main">
    <w:body>
        <w:p w:rsidR="005F670F" w:rsidRDefault="005F79F5">
            <w:r>
                <w:t>Test</w:t>
            </w:r>
            <w:bookmarkStart w:id="0" w:name="_GoBack"/>
            <w:bookmarkEnd w:id="0"/>
        </w:p>
        <w:sectPr w:rsidR="005F670F">
            <w:pgSz w:w="12240" w:h="15840"/>
            <w:pgMar w:top="1440" w:right="1440" w:bottom="1440" w:left="1440"
                     w:header="720" w:footer="720" w:gutter="0"/>
            <w:cols w:space="720"/>
            <w:docGrid w:linePitch="360"/>
        </w:sectPr>
    </w:body>
</w:document>
```

Here:

- <w:document> - document itself
- <w:body> - document body
- <w:p> - paragraph
- <w:r> - run (fragment) of the text
- <w:t> - text itself
- <w:sectPr> - page description

When you open this document in a text editor, you will see a document with a single word Test.

word/rels/document.xml.rels

Contains a list of links of word/document.xml part. Name of a link file is created from the filename of the document part, to which it belongs, with rels extension. A folder with link file is called rels, and it is placed at the same level as a part to which it relates. There are no links in word/document.xml, so the file is empty:

```
<Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
</Relationships>
```

Even if there are no links, this file must exist.

docx and Microsoft Word

docx created with Microsoft Word or any other editor contains several additional files.

These files are:

- docProps/core.xml - the basic document metadata according to Open Packaging Conventions and Dublin Core [1], [2].
- docProps/app.xml - general information about the document: number of pages, words, characters, application in which document was created, etc.
- word/settings.xml - settings for the document.
- word/styles.xml - styles applied to the document. Separates data from view.
- word/webSettings.xml - HTML display settings and document conversion settings to HTML.
- word/fontTable.xml - list of document fonts.
- word/theme1.xml - theme (consists of color schemes, fonts, and formatting).

Complex documents usually have more parts.

Reverse engineering docx

The initial task is to find out how any document fragment is stored in xml, then to create (or parse) such documents on our own. We need:

- Zip archiver
- Library for XML formatting (Word gives XML without indents, all in one line)
- A tool for viewing diff between files, I use git and TortoiseGit

Tools

- For Windows: zip, unzip, libxml2, git, TortoiseGit

- For Linux: apt-get install zip unzip libxml2 libxml2-utils git

Also, scripts will be necessary for automatic archiving and XML formatting. Usage on Windows:

- unpack file dir - unpacks document file in folder dir and formats xml
- pack dir file - pack folder dir into the document file

Usage on Linux is similar, but use ./unpack.sh instead of unpack, and pack becomes ./pack.

Use

To search for changes:

1. Create an empty docx file in the editor.
2. Unpack it using unpack into the new folder.
3. Commit new folder.
4. Add things you need (hyperlink, table, etc.) to the document from step 1.
5. Unpack modified file into an existing folder.
6. Explore diff, removing unnecessary changes (links permutation, namespace order, etc.).
7. Pack folder into document and check that it opens.
8. Commit changed folder.

Example 1. Bold text

As a first example we'll search for a tag that makes text bold.

1. Create bold.docx document with normal (not bold) text Test.
2. Unpack it: unpack bold.docx bold.
3. Commit the result.
4. Make Test text bold.
5. Unpack it: unpack bold.docx bold.
6. Initially, the diff was as follows:

Let's examine it in detail:

docProps/app.xml

```
@@ -1,9 +1,9 @@
-  <TotalTime>0</TotalTime>
+  <TotalTime>1</TotalTime>
```

The time change is not necessary.

docProps/core.xml

```
@@ -4,9 +4,9 @@
-  <cp:revision>1</cp:revision>
```

```
+  <cp:revision>2</cp:revision>
   <dcterms:created xsi:type="dcterms:W3CDTF">2017-02-07T19:37:00Z</dcterms:created>
-  <dcterms:modified xsi:type="dcterms:W3CDTF">2017-02-07T19:37:00Z</dcterms:modified
>
+  <dcterms:modified xsi:type="dcterms:W3CDTF">2017-02-08T10:01:00Z</dcterms:modified
>
```

Document version and modification date changes are irrelevant.

word/document.xml

```
@@ -1,24 +1,26 @@
    <w:body>
-    <w:p w:rsidR="0076695C" w:rsidRPr="00290C70" w:rsidRDefault="00290C70">
+    <w:p w:rsidR="0076695C" w:rsidRPr="00F752CF" w:rsidRDefault="00290C70">
       <w:pPr>
         <w:rPr>
+          <w:b/>
           <w:lang w:val="en-US"/>
         </w:rPr>
       </w:pPr>
-      <w:r>
+      <w:r w:rsidRPr="00F752CF">
         <w:rPr>
+          <w:b/>
           <w:lang w:val="en-US"/>
         </w:rPr>
         <w:t>Test</w:t>
       </w:r>
       <w:bookmarkStart w:id="0" w:name="_GoBack"/>
       <w:bookmarkEnd w:id="0"/>
     </w:p>
-    <w:sectPr w:rsidR="0076695C" w:rsidRPr="00290C70">
+    <w:sectPr w:rsidR="0076695C" w:rsidRPr="00F752CF">
```

Changes in w:rsidR are unnecessary - it is utility information for Microsoft Word. The main change here:

```
         <w:rPr>
+          <w:b/>
```

in the paragraph with Test. Apparently element <w:b/> makes the text bold. Let's save this change and revert the rest.

word/settings.xml

```
@@ -1,8 +1,9 @@
+  <w:proofState w:spelling="clean"/>
@@ -17,10 +18,11 @@
+    <w:rsid w:val="00F752CF"/>
```

It does not contain anything related to the bold text. Revert.

7 Pack a folder with one relevant change (adding <w:b/>) and check that document opens and shows as expected.
8 Commit the change.

Example 2. Footer

Let's move on to a more complex example - adding a footer to a document. Initial commit. Add footer text ‘ 123 ’ and unpack the document. Initial diff looks like this:

Immediately revert changes in docProps/app.xml and docProps/core.xml – same as with the first example.

[Content_Types].xml

```
@@ -4,10 +4,13 @@
   <Default Extension="xml" ContentType="application/xml"/>
   <Override PartName="/word/document.xml" ContentType="application/vnd.openxmlformat
s-officedocument.wordprocessingml.document.main+xml"/>
+  <Override PartName="/word/footnotes.xml" ContentType="application/vnd.openxmlforma
ts-officedocument.wordprocessingml.footnotes+xml"/>
+  <Override PartName="/word/endnotes.xml" ContentType="application/vnd.openxmlformat
s-officedocument.wordprocessingml.endnotes+xml"/>
+  <Override PartName="/word/footer1.xml" ContentType="application/vnd.openxmlformats-
officedocument.wordprocessingml.footer+xml"/>
```

footer clearly looks like what we need, but what should we do with footnotes and endnotes? Are they required for adding a footer, or are the just a byproduct created at the same time? Discerning the answer is not always easy, here are the main approaches:

- Explore the changes: are they connected with each other?
- Experiment
- Finally, if you still don't understand what's going on:

word/_rels/document.xml.rels

Initial diff looks like this:

```
@@ -1,8 +1,11 @@
 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
 <Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
+  <Relationship Id="rId5" Type="http://schemas.openxmlformats.org/officeDocument/200
6/relationships/theme" Target="theme/theme1.xml"/>
   <Relationship Id="rId3" Type="http://schemas.openxmlformats.org/officeDocument/200
6/relationships/webSettings" Target="webSettings.xml"/>
+  <Relationship Id="rId4" Type="http://schemas.openxmlformats.org/officeDocument/200
6/relationships/fontTable" Target="fontTable.xml"/>
   <Relationship Id="rId2" Type="http://schemas.openxmlformats.org/officeDocument/200
6/relationships/settings" Target="settings.xml"/>
   <Relationship Id="rId1" Type="http://schemas.openxmlformats.org/officeDocument/200
6/relationships/styles" Target="styles.xml"/>
-  <Relationship Id="rId5" Type="http://schemas.openxmlformats.org/officeDocument/200
6/relationships/theme" Target="theme/theme1.xml"/>
-  <Relationship Id="rId4" Type="http://schemas.openxmlformats.org/officeDocument/200
6/relationships/fontTable" Target="fontTable.xml"/>
+  <Relationship Id="rId6" Type="http://schemas.openxmlformats.org/officeDocument/200
6/relationships/footer" Target="footer1.xml"/>
+  <Relationship Id="rId7" Type="http://schemas.openxmlformats.org/officeDocument/200
6/relationships/endnotes" Target="endnotes.xml"/>
```

```
+  <Relationship Id="rId8" Type="http://schemas.openxmlformats.org/officeDocument/200
6/relationships/footnotes" Target="footnotes.xml"/>
 </Relationships>
```

As we see some of the changes are due to the fact that Word has changed links order, let's restore the order and make diff smaller:

```
@@ -3,6 +3,9 @@
+  <Relationship Id="rId6" Type="http://schemas.openxmlformats.org/officeDocument/200
6/relationships/footer" Target="footer1.xml"/>
+  <Relationship Id="rId7" Type="http://schemas.openxmlformats.org/officeDocument/200
6/relationships/endnotes" Target="endnotes.xml"/>
+  <Relationship Id="rId8" Type="http://schemas.openxmlformats.org/officeDocument/200
6/relationships/footnotes" Target="footnotes.xml"/>
```

footer, footnotes, endnotes appear again. All of them are connected with the main document, let's have a look at it:

word/document.xml

```
@@ -15,10 +15,11 @@
       </w:r>
       <w:bookmarkStart w:id="0" w:name="_GoBack"/>
       <w:bookmarkEnd w:id="0"/>
     </w:p>
     <w:sectPr w:rsidR="0076695C" w:rsidRPr="00290C70">
+      <w:footerReference w:type="default" r:id="rId6"/>
       <w:pgSz w:w="11906" w:h="16838"/>
       <w:pgMar w:top="1134" w:right="850" w:bottom="1134" w:left="1701" w:header="70
8" w:footer="708" w:gutter="0"/>
       <w:cols w:space="708"/>
       <w:docGrid w:linePitch="360"/>
     </w:sectPr>
```

For a change, there are only necessary changes – an explicit link to the footer in sectPr. There are no links to footnotes and endnotes in the document, so we can assume they are not necessary.

word/settings.xml

```
@@ -1,19 +1,30 @@
 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
 <w:settings xmlns:w="http://schemas.openxmlformats.org/wordprocessingml/2006/main">
   <w:zoom w:percent="100"/>
+  <w:proofState w:spelling="clean"/>
   <w:defaultTabStop w:val="708"/>
   <w:characterSpacingControl w:val="doNotCompress"/>
+  <w:footnotePr>
+    <w:footnote w:id="-1"/>
+    <w:footnote w:id="0"/>
+  </w:footnotePr>
+  <w:endnotePr>
+    <w:endnote w:id="-1"/>
+    <w:endnote w:id="0"/>
+  </w:endnotePr>
```

```
   <w:compat>
     <w:compatSetting w:name="compatibilityMode" w:uri="http://schemas.microsoft.com/
office/word" w:val="15"/>
     <w:compatSetting w:name="overrideTableStyleFontSizeAndJustification" w:uri="http
://schemas.microsoft.com/office/word" w:val="1"/>
     <w:compatSetting w:name="enableOpenTypeFeatures" w:uri="http://schemas.microsoft
.com/office/word" w:val="1"/>
     <w:compatSetting w:name="doNotFlipMirrorIndents" w:uri="http://schemas.microsoft
.com/office/word" w:val="1"/>
     <w:compatSetting w:name="differentiateMultirowTableHeaders" w:uri="http://schema
s.microsoft.com/office/word" w:val="1"/>
   </w:compat>
   <w:rsids>
     <w:rsidRoot w:val="00290C70"/>
+    <w:rsid w:val="000A7B7B"/>
+    <w:rsid w:val="001B0DE6"/>
```

Settings lists links to footnotes and endnotes which presumably adds them to the document. Note that footer does not appear here.

word/styles.xml

```
@@ -480,6 +480,50 @@
       <w:rFonts w:ascii="Times New Roman" w:hAnsi="Times New Roman"/>
       <w:b/>
       <w:sz w:val="28"/>
     </w:rPr>
   </w:style>
+  <w:style w:type="paragraph" w:styleId="a4">
+    <w:name w:val="header"/>
+    <w:basedOn w:val="a"/>
+    <w:link w:val="a5"/>
+    <w:uiPriority w:val="99"/>
+    <w:unhideWhenUsed/>
+    <w:rsid w:val="000A7B7B"/>
+    <w:pPr>
+      <w:tabs>
+        <w:tab w:val="center" w:pos="4677"/>
+        <w:tab w:val="right" w:pos="9355"/>
+      </w:tabs>
+      <w:spacing w:after="0" w:line="240" w:lineRule="auto"/>
+    </w:pPr>
+  </w:style>
+  <w:style w:type="character" w:customStyle="1" w:styleId="a5">
+    <w:name w:val="??????? ?????????? ????"/>
+    <w:basedOn w:val="a0"/>
+    <w:link w:val="a4"/>
+    <w:uiPriority w:val="99"/>
+    <w:rsid w:val="000A7B7B"/>
+  </w:style>
+  <w:style w:type="paragraph" w:styleId="a6">
+    <w:name w:val="footer"/>
+    <w:basedOn w:val="a"/>
+    <w:link w:val="a7"/>
+    <w:uiPriority w:val="99"/>
+    <w:unhideWhenUsed/>
+    <w:rsid w:val="000A7B7B"/>
```

```
+    <w:pPr>
+      <w:tabs>
+        <w:tab w:val="center" w:pos="4677"/>
+        <w:tab w:val="right" w:pos="9355"/>
+      </w:tabs>
+      <w:spacing w:after="0" w:line="240" w:lineRule="auto"/>
+    </w:pPr>
+  </w:style>
+  <w:style w:type="character" w:customStyle="1" w:styleId="a7">
+    <w:name w:val="?????? ?????????? ????"/>
+    <w:basedOn w:val="a0"/>
+    <w:link w:val="a6"/>
+    <w:uiPriority w:val="99"/>
+    <w:rsid w:val="000A7B7B"/>
+  </w:style>
 </w:styles>
```

We are interested in style changes, only if we are looking for style changes. In this case, the change can be reverted.

word/footer1.xml

Take a look at footer itself (some namespaces are omitted for readability, but they should be present in the document):

```
<w:ftr xmlns:w="http://schemas.openxmlformats.org/wordprocessingml/2006/main">
  <w:p w:rsidR="000A7B7B" w:rsidRDefault="000A7B7B">
    <w:pPr>
      <w:pStyle w:val="a6"/>
    </w:pPr>
    <w:r>
      <w:t>123</w:t>
    </w:r>
  </w:p>
</w:ftr>
```

Here is the footer text: ‘ 123 ’ . As we don't track style changes in this example we need to remove the link to <w:pStyle w:val="a6"/>.

The analysis of all the changes results in the following assumptions:

- Fotnotes and endnotes are unnecessary
- We need to add word/footer1.xml file
- In [Content_Types].xml we need to add a link to footer
- In word/_rels/document.xml.rels we need to add a link to footer
- In word/document.xml to <w:sectPr> tag we need to add <w:footerReference>

These assumptions reduce the diff to this set of changes:

Then pack document and open it. If everything was done correctly, the document will open and there will be a footer with text ‘ 123 ’ . And here is the final commit.

Thus, the process of change detection is reduced to determining a minimum set of changes sufficient to achieve

the desired result.

Practice

After we find the necessary change, it is logical to proceed to the next stage, it could be any of:

- Creating docx
- Parsing docx
- Converting docx

And for that, we need XSLT and XPath.

Let's write a fairly simple conversion - replacement or addition of footer for a document.

Algorithm

Algorithm looks like this:

1. Unpack the document
2. Add our footer
3. Add a link to it to [ContentTypes].xml and word/rels/document.xml.rels
4. In word/document.xml in each <w:sectPr> tag add or replace <w:footerReference> tag with reference to our footer
5. Pack the document.

Let's start.

Unpacking

In Caché ObjectScript it is possible to execute operating system commands using the function $zf(-1, oscommand). Call unzip to unpack the document using wrapper over $zf(-1):

```
/// Using %3 (unzip) unpack file %1 in folder %2
Parameter UNZIP = "%3 %1 -d %2";

/// Unpack archive source in folder targetDir
ClassMethod executeUnzip(source, targetDir) As %Status
{
    set timeout = 100
    set cmd = $$$FormatText(..#UNZIP, source, targetDir, ..getUnzip())
    return ..execute(cmd, timeout)
}
```

Creation of footer file

Input receives the footer text, we will write it to in.xml file:

```
<xml>TEST</xml>
```

In XSLT (file footer.xsl) we will create a footer with text from xml tag (some namespaces are omitted, here is the full list):

```
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns="http://schemas.openxmlformats.org/package/2006/relationships" version="1.0">
    <xsl:output method="xml" omit-xml-declaration="no" indent="yes"
standalone="yes"/>
    <xsl:template match="/">

        <w:ftr xmlns:w="
http://schemas.openxmlformats.org/wordprocessingml/2006/main">
            <w:p>
                <w:r>
                    <w:rPr>
                        <w:lang w:val="en-US"/>
                    </w:rPr>
                    <w:t>
                        <xsl:value-of select="//xml/text()"/>
                    </w:t>
                </w:r>
            </w:p>
        </w:ftr>
    </xsl:template>
</xsl:stylesheet>
```

Call XSLT converter:

```
do ##class(%XML.XSLT.Transformer).TransformFile("in.xml", "footer.xsl", footer0.xml")
```

The result is the footer file footer0.xml:

```
<w:ftr xmlns:w="http://schemas.openxmlformats.org/wordprocessingml/2006/main">
    <w:p>
        <w:r>
            <w:rPr>
                <w:lang w:val="en-US"/>
            </w:rPr>
            <w:t>TEST</w:t>
        </w:r>
    </w:p>
</w:ftr>
```

Add a footer link to a list of links of the main document

The link with rId0 ID usually does not exist. However, you can use XPath to get the ID which does not exist. Add a link to footer0.xml with rId0 ID in word/rels/document.xml.rels:

```
<xsl:stylesheet  xmlns:xsl="
http://www.w3.org/1999/XSL/Transform"  xmlns="
http://schemas.openxmlformats.org/package/2006/relationships"  version="1.0">
    <xsl:output method="xml" omit-xml-declaration="yes" indent="no"  />
```

```
    <xsl:param name="new">
        <Relationship
           Id="rId0"
           Type="
http://schemas.openxmlformats.org/officeDocument/2006/relationships/footer"
           Target="footer0.xml"/>
    </xsl:param>

    <xsl:template match="/*">
        <xsl:copy>
            <xsl:copy-of select="$new"/>
            <xsl:copy-of select="@* | node()"/>
        </xsl:copy>
    </xsl:template>
</xsl:stylesheet>
```

Specify links in document

Next, it is necessary in each <w:sectPr> tag to add <w:footerReference> tag or replace a link in it with the link to our footer. It turns out that each <w:sectPr> tag can have 3 different <w:footerReference> tags - for the first page, even pages and the rest:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships"
xmlns:w="http://schemas.openxmlformats.org/wordprocessingml/2006/main"
version="1.0">
    <xsl:output method="xml" omit-xml-declaration="yes" indent="yes" />
    <xsl:template match="//@* | //node()">
        <xsl:copy>
            <xsl:apply-templates select="@*"/>
            <xsl:apply-templates select="node()"/>
        </xsl:copy>
    </xsl:template>
    <xsl:template match="//w:sectPr">
        <xsl:element name="{name()}" namespace="{namespace-uri()}">
            <xsl:copy-of select="./namespace::*"/>
            <xsl:apply-templates select="@*"/>
            <xsl:copy-of select="./*[local-name() != 'footerReference']"/>
            <w:footerReference w:type="default" r:id="rId0"/>
            <w:footerReference w:type="first" r:id="rId0"/>
            <w:footerReference w:type="even" r:id="rId0"/>
        </xsl:element>
    </xsl:template>
</xsl:stylesheet>
```

Add footer in [Content_Types].xml

In [Content_Types].xml add information that /word/footer0.xml is a application/vnd.openxmlformats-officedocument.wordprocessingml.footer+xml:

```
<xsl:stylesheet  xmlns:xsl="
http://www.w3.org/1999/XSL/Transform"  xmlns="
http://schemas.openxmlformats.org/package/2006/content-types"  version="1.0">
    <xsl:output method="xml" omit-xml-declaration="yes" indent="no"  />
    <xsl:param name="new">
```

```
        <Override
         PartName="/word/footer0.xml"
         ContentType="application/vnd.openxmlformats-
officedocument.wordprocessingml.footer+xml"/>
    </xsl:param>

    <xsl:template match="/*">
        <xsl:copy>
            <xsl:copy-of select="@* | node()"/>
            <xsl:copy-of select="$new"/>
        </xsl:copy>
    </xsl:template>
</xsl:stylesheet>
```

As a result

Full code is available on GitHub. It works like this:

```
do ##class(Converter.Footer).modifyFooter("in.docx", "out.docx", "TEST")
```

Where:

- in.docx - original document
- out.docx - output document
- TEST - text which is added to footer

Conclusions

Using only XSLT and ZIP, you can successfully work with docx documents, xlsx tables, and pptx presentations.

Open questions

1. Do you generate or parse xlsx, docx? If so, how?
2. Iṁ looking for XSD files with schemas of ECMA-367 of version 5 and comments. The XSD of the fifth version is available for download on ECMA site. But it is difficult to comprehend it without any comments. The XSD of the second version is available with comments.

Links

- ECMA-376
- docx description
- Detailed article about docx
- Repository with scripts
- Repository with footer editor

#Caché #XML