Article
Rizmaan Marikar · Dec 25, 2021 13m read

Generating EXCEL documents with Caché ObjectScript

There are many ways to generate excel files using Intersystems, some of them are ZEN reports, IRIS reports (Logi reports or formally known as JReports), or we can use third party Java libraries, the possibilities are almost endless.

But, what if you want to create a simple spreadsheet with only Caché ObjectScript? (no third party applications)

In my case i need to generate reports that contains large raw data (the financial guys love them), but my ZEN/IRIS fails, gives me, what i would like to call a "zero byte file", basically says java ran out of memory, and causes heavy load on the reporting server.

This can be done using Office Open XML (OOXML). The Office Open XML format is comprised of a number of XML files within a ZIP package. So basically we need to generate these XML files and zip them rename it to .xslx. It's that simple.

The files follow a simple set of conventions called the open packaging conventions. You need to declare the content types of the parts, as well as tell the consuming application where it should start.

In order to create a simple spreadsheet we need a minimum of 5 files;

- workbook.xml
- worksheet.xml
- [ContentTypes].xml
- styles.xml
- rels
    - .rels
    - workbook.xml.rels

workbook.xml
The workbook is the container for the various worksheets. The workbook is where you can reference the styles part, shared string tables, and any other pieces of information that apply to the entire Spreadsheet file.

```
ClassMethod GenerateWorkbookXML(){
    set status =$$$OK
    set xmlfile = tempDirectoryPath_"workbook.xml"
    try{
    set stream = ##class(%Stream.FileCharacter).%New()
    set sc=stream.LinkToFile(xmlfile)
    do stream.WriteLine("<?xml version='1.0' encoding='UTF-8' standalone='yes'?>")
    do stream.WriteLine("<workbook xmlns='http://schemas.openxmlformats.org/spreadshe
etml/2006/main' xmlns:r='http://schemas.openxmlformats.org/officeDocument/2006/relati
onships'>")
    do
stream.WriteLi
ne("<sheets> <sheet name='"_workSheetName_"' sheetId='1' r:id='rId1'/>")
    do stream.WriteLine("</sheets> </workbook>")

        do stream.%Save()
```

```
    }catch{
        set status=$$$NO
    }
    kill stream
    return status
}
```

rels/workbook.xml.rels

We just need to create a relationship that has an id of rId1 so that it will match the reference from the workbook.xml part

```
ClassMethod CreateRelsXML(){
set status =$$$OK

    set isunix=$zcvt($p($zv," ",3,$l($p($zv," ("),")")),"U")["UNIX"
    if isunix {
        set ext="/"
    }else{
        set ext="\"
    }
    set xmlfile = fileDirectory_"_rels"_ext_"workbook.xml.rels"
    set stream = ##class(%Stream.FileCharacter).%New()
    set sc=stream.LinkToFile(xmlfile)
    do stream.WriteLine("<?xml version='1.0' encoding='UTF-8' standalone='yes'?>")
    do stream.WriteLine("<Relationships xmlns='http://schemas.openxmlformats.org/pack
age/2006/relationships'>")
      do stream.WriteLine("<Relationship Id='rId1'
Type='http://schemas.openxmlformats.org/officeDocument/2006/relationships/worksheet'
Target='worksheet.xml'/>")
      do stream.WriteLine("<Relationship Id='rId2' Type='http://schemas.openxmlformat
s.org/officeDocument/2006/relationships/styles' Target='styles.xml' />")
    do stream.WriteLine("</Relationships>")
    try{
        do stream.%Save()
    }catch{
        set status=$$$NO
    }
    kill stream
    set xmlfile = fileDirectory_"_rels"_ext_".rels"
    set stream = ##class(%Stream.FileCharacter).%New()
    set sc=stream.LinkToFile(xmlfile)

    do stream.WriteLine("<?xml version='1.0' encoding='UTF-8' standalone='yes'?>")
    do stream.WriteLine("<Relationships xmlns='http://schemas.openxmlformats.org/pack
age/2006/relationships'>")
      do stream.WriteLine("<Relationship Id='rId1' Type='http://schemas.openxmlformat
s.org/officeDocument/2006/relationships/officeDocument' Target='workbook.xml'/>")
    do stream.WriteLine("</Relationships>")
    try{
        do stream.%Save()
    }catch{
        set status=$$$NO
    }
    kill stream
    return status
}
```

[ContentTypes].xml

Static file(at the moment, it should be a dynamic file depending on the number of worksheets) links workbook worksheet and stypes together.Every Office Open XML file must declare the content types used in the ZIP package. That is done with the [Content_Types].xml file.

```
ClassMethod GenerateConntentTypesXML(){
    set status =$$$OK
    set xmlfile = tempDirectoryPath_"[Content_Types].xml"
    set stream = ##class(%Stream.FileCharacter).%New()
    set sc=stream.LinkToFile(xmlfile)
    try{
        do stream.WriteLine("<?xml version='1.0' encoding='UTF-8' standalone='yes'?>"
)
        do stream.WriteLine("<Types xmlns='http://schemas.openxmlformats.org/package/
2006/content-types'>")
          do stream.WriteLine("<Default Extension='rels' ContentType='application/vnd
.openxmlformats-package.relationships+xml'/>")
        do stream.WriteLine("<Override PartName='/workbook.xml' ContentType='applicat
ion/vnd.openxmlformats-officedocument.spreadsheetml.sheet.main+xml'/>")
          do stream.WriteLine("<Override PartName='/worksheet.xml' ContentType='appli
cation/vnd.openxmlformats-officedocument.spreadsheetml.worksheet+xml'/>")
          do stream.WriteLine("<Override PartName='/styles.xml' ContentType='applicat
ion/vnd.openxmlformats-officedocument.spreadsheetml.styles+xml' />")
        do stream.WriteLine("</Types>")
        do stream.%Save()
    }catch{
        set status=$$$NO
    }
    kill stream
    return status
}
```

styles.xml

All the formatting sits here, at the moment i have some static styles added, (planning to convert it to a more dynamic workbook specific)

Excel Styles	ID	Style	Excel Format
	1	default	Text
	2	#;[Red]-#	Number
	3	#.##;[Red]-#.##	Number
	4	yyyy/mm/dd	Date
	5	hh:mm	Date
	6	Header and Center Aligned	Text
	7	Header 2 Left Aligned	Text
	8	Good(Green Highlight)	General
	9	Bad(Red Highlight)	General
	10	Neutral(Orange Highlight)	General
	11	yyyy/mm/dd hh:mm	Date

```
ClassMethod CreateStylesXML(){
    set status =$$$OK
    set xmlfile = tempDirectoryPath_"styles.xml"
    try{
```

```
        set stream = ##class(%Stream.FileCharacter).%New()
        set sc=stream.LinkToFile(xmlfile)
         do stream.WriteLine("<?xml version=""1.0"" encoding=""UTF-8"" standalone=""y
es""?>")
        do stream.WriteLine("<styleSheet xmlns=""http://schemas.openxmlformats.org/sp
readsheetml/2006/main"" xmlns:mc=""http://schemas.openxmlformats.org/markup-compatibi
lity/2006"" mc:Ignorable=""x14ac x16r2 xr"" xmlns:x14ac=""http://schemas.microsoft.co
m/office/spreadsheetml/2009/9/ac"" xmlns:x16r2=""http://schemas.microsoft.com/office/
spreadsheetml/2015/02/main"" xmlns:xr=""http://schemas.microsoft.com/office/spreadshe
etml/2014/revision"">")
        do stream.WriteLine("<numFmts count=""4"">")
        do stream.WriteLine("<numFmt numFmtId=""166"" formatCode=""#,##0;[Red]\-#,##0
""/>")
        do stream.WriteLine("<numFmt numFmtId=""168"" formatCode=""#,##0.00;[Red]\-#,
##0.00""/>")
        do stream.WriteLine("<numFmt numFmtId=""169"" formatCode=""dd\/mm\/yyyy;@""/>
")
        do stream.WriteLine("<numFmt numFmtId=""170"" formatCode=""dd/mm/yyyy\ hh:mm"
"/></numFmts>")
        do stream.WriteLine("<fonts count=""5"" x14ac:knownFonts=""1"">")
        do stream.WriteLine("<font><sz val=""10""/><color theme=""1""/><name val=""Ca
libri""/><family val=""2""/><scheme val=""minor""/></font>")
        do stream.WriteLine("<font><sz val=""10""/><color rgb=""FF006100""/><name val
=""Calibri""/><family val=""2""/><scheme val=""minor""/></font>")
        do stream.WriteLine("<font><sz val=""10""/><color rgb=""FF9C0006""/><name val
=""Calibri""/><family val=""2""/><scheme val=""minor""/></font>")
        do stream.WriteLine("<font><sz val=""10""/><color rgb=""FF9C5700""/><name val
=""Calibri""/><family val=""2""/><scheme val=""minor""/></font>")
        do stream.WriteLine("<font><b/><sz val=""10""/><color theme=""1""/><name val=
""Calibri""/><family val=""2""/><scheme val=""minor""/></font></fonts>")
        do stream.WriteLine("<fills count=""5"">")
        do stream.WriteLine("<fill><patternFill patternType=""none""/></fill>")
        do stream.WriteLine("<fill><patternFill patternType=""gray125""/></fill>")
        do stream.WriteLine("<fill><patternFill patternType=""solid""><fgColor rgb=""
FFC6EFCE""/></patternFill></fill>")
        do stream.WriteLine("<fill><patternFill patternType=""solid""><fgColor rgb=""
FFFFC7CE""/></patternFill></fill>")
        do stream.WriteLine("<fill><patternFill patternType=""solid""><fgColor rgb=""
FFFFEB9C""/></patternFill></fill></fills>")
        do stream.WriteLine("<borders count=""1""><border><left/><right/><top/><botto
m/><diagonal/></border></borders>")
        do stream.WriteLine("<cellStyleXfs count=""4"">")
        do stream.WriteLine("<xf numFmtId=""0"" fontId=""0"" fillId=""0"" borderId=""
0""/>")
        do stream.WriteLine("<xf numFmtId=""0"" fontId=""1"" fillId=""2"" borderId=""
0"" applyNumberFormat=""0"" applyBorder=""0"" applyAlignment=""0"" applyProtection=""
0""/>")
        do stream.WriteLine("<xf numFmtId=""0"" fontId=""2"" fillId=""3"" borderId=""
0"" applyNumberFormat=""0"" applyBorder=""0"" applyAlignment=""0"" applyProtection=""
0""/>")
        do stream.WriteLine("<xf numFmtId=""0"" fontId=""3"" fillId=""4"" borderId=""
0"" applyNumberFormat=""0"" applyBorder=""0"" applyAlignment=""0"" applyProtection=""
0""/></cellStyleXfs>")
        do stream.WriteLine("<cellXfs count=""12""><xf numFmtId=""0"" fontId=""0"" fi
llId=""0"" borderId=""0"" xfId=""0""/>")
        do stream.WriteLine("<xf numFmtId=""49"" fontId=""0"" fillId=""0"" borderId="
"0"" xfId=""0"" quotePrefix=""1"" applyNumberFormat=""1""/>")
        do stream.WriteLine("<xf numFmtId=""166"" fontId=""0"" fillId=""0"" borderId=
""0"" xfId=""0"" applyNumberFormat=""1""/>")
```

```
        do stream.WriteLine("<xf numFmtId=""168"" fontId=""0"" fillId=""0"" borderId=
""0"" xfId=""0"" applyNumberFormat=""1""/>")
        do stream.WriteLine("<xf numFmtId=""169"" fontId=""0"" fillId=""0"" borderId=
""0"" xfId=""0"" applyNumberFormat=""1""/>")
        do stream.WriteLine("<xf numFmtId=""20"" fontId=""0"" fillId=""0"" borderId="
"0"" xfId=""0"" applyNumberFormat=""1""/>")
        do stream.WriteLine("<xf numFmtId=""49"" fontId=""4"" fillId=""0"" borderId="
"0"" xfId=""0"" applyNumberFormat=""1"" applyFont=""1""/>")
        do stream.WriteLine("<xf numFmtId=""49"" fontId=""4"" fillId=""0"" borderId="
"0"" xfId=""0"" applyNumberFormat=""1"" applyFont=""1"" applyAlignment=""1""><alignme
nt horizontal=""center""/>")
        do stream.WriteLine("</xf>")
        do stream.WriteLine("<xf numFmtId=""49"" fontId=""1"" fillId=""2"" borderId="
"0"" xfId=""1"" applyNumberFormat=""1""/>")
        do stream.WriteLine("<xf numFmtId=""0"" fontId=""2"" fillId=""3"" borderId=""
0"" xfId=""2""/>")
        do stream.WriteLine("<xf numFmtId=""0"" fontId=""3"" fillId=""4"" borderId=""
0"" xfId=""3""/>")
        do stream.WriteLine("<xf numFmtId=""170"" fontId=""0"" fillId=""0"" borderId=
""0"" xfId=""0"" applyNumberFormat=""1""/></cellXfs>")
        do stream.WriteLine("<cellStyles count=""4""><cellStyle name=""Bad"" xfId=""2
"" builtinId=""27""/>")
        do stream.WriteLine("<cellStyle name=""Good"" xfId=""1"" builtinId=""26""/><c
ellStyle name=""Neutral"" xfId=""3"" builtinId=""28""/>")
        do stream.WriteLine("<cellStyle name=""Normal"" xfId=""0"" builtinId=""0""/><
/cellStyles><dxfs count=""0""/>")
        do stream.WriteLine("<tableStyles count=""0"" defaultTableStyle=""TableStyleM
edium2"" defaultPivotStyle=""PivotStyleLight16""/>    ")
        do stream.WriteLine("<extLst><ext uri=""{EB79DEF2-80B8-43e5-95BD-54CBDDF9020C
}"" xmlns:x14=""http://schemas.microsoft.com/office/spreadsheetml/2009/9/main"">")
        do stream.WriteLine("<x14:slicerStyles defaultSlicerStyle=""SlicerStyleLight1
""/></ext><ext uri=""{9260A510-F301-46a8-8635-F512D64BE5F5}"" xmlns:x15=""http://sche
mas.microsoft.com/office/spreadsheetml/2010/11/main"">")
        do stream.WriteLine("<x15:timelineStyles defaultTimelineStyle=""TimeSlicerSty
leLight1""/></ext></extLst>")
        do stream.WriteLine("</styleSheet>")
        do stream.%Save()
    }catch{
        set status=$$$NO
    }
    kill stream
    return status
}
```

worksheet.xml

This is where our data sits. The first row in the sheet will have the column titles. The next rows will only have data in the first columns.
We will define the column widths for each column here, if not by default columns will be set to autofit.

Sample worksheet xml

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<worksheet xmlns="https://schemas.openxmlformats.org/spreadsheetml/2006/main" xmlns:r
="https://schemas.openxmlformats.org/officeDocument/2006/relationships">
<sheetData>
<row>
    <c t="inlineStr">
        <is>
```

```
            <t>Name</t>
        </is>
    </c>
    <c t="inlineStr">
        <is>
            <t>Amount</t>
        </is>
    </c>
</row>
<row>
    <c t="inlineStr">
        <is>
            <t>Jhon Smith</t>
        </is>
    </c>
    <c>
        <v>1000.74</v>
    </c>
</row>
<row>
    <c t="inlineStr">
        <is>
            <t>Tracy A</t>
        </is>
    </c>
    <c>
        <v>6001.74</v>
    </c>
</row>
</sheetData>
</worksheet>
```

Sample Excel

A	B
Name	Amount
Jhon Smith	1000.74
Tracy A	6001.74

Formulas within the worksheet can we done using a function <f> tag

```
<c >
<f>B2*0.08</f >
</c >
<c >
```

```
<f>B2+C2</f >
</c>
```

and finally we zip them, rename it to.xlsx (using unix zip)

```
set cmd ="cd "_fileDirectory_" && find . -type f | xargs zip .."_ext_xlsxFile
```

Generating an excel Document.

the following sample code generates an excel document.

```
set file = "/temp/test.xlsx"
set excelObj = ##class(XLSX.writer).%New(file)
do excelObj.SetWorksheetName("test1")
set status = excelObj.BeginWorksheet()
set row = 0
set row = row+1
;----------- excelObj.Cells(rowNumber,columnNumber,style,content)

set status = excelObj.Cells(row,1,1,"Header1")
set row = row+1
set status = excelObj.Cells(row,1,2,"Content 1")
set status = excelObj.EndWorksheet()
W !,excelObj.fileName
```

Excel Writer Class can be found here xlsx.writer.xml.zip

#ObjectScript #Caché #InterSystems IRIS #InterSystems IRIS for Health