

記事

[Toshihiko Minamoto](#) · 2022年3月8日 8m read

Caché ObjectScriptによるExcelドキュメントの生成

InterSystemsを使用してExcelファイルを生成する方法はたくさんあります。ZENレポートやIRISレポート（Logiレポートまたは正式にはJReportsと呼ばれるレポート）のほか、サードパーティのJavaライブラリを使用するなど、可能性はほぼ無限です。

しかし、*Caché ObjectScript*だけで単純なスプレッドシートを作成したい場合はどうでしょうか。（サードパーティアプリケーションを使用せずに、です）

私の場合、大量の生データを含むレポート（金融関係の人たちが好むレポート）を生成する必要がありますが、私のZEN/IRISでは対応できません。私が呼ぶところの「ゼロバイトファイル」が生成され、基本的にJavaのメモリ不足となり、レポートングサーバーに大きな負荷を生じてしまいます。

これは、Office Open XML（OOXML）を使って実現できます。Office Open XML形式は、多数のXMLファイルで構成されるZIPパッケージです。つまり基本的には、これらのXMLファイルを生成してZIP圧縮し、.xlsxに名前を変更すればよいのです。それくらい単純です。

ファイルは、Open Packaging Conventionsという単純な命名規則に従っています。パーツのコンテンツタイプを宣言し、消費するアプリケーションにどこから開始するかを指示する必要があります。

単純なスプレッドシートを作成するには、少なくとも5つのファイルが必要です。

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

workbook.xml

workbookは、様々なワークシートをまとめるコンテナです。workbookでは、スタイルパーツ、共有文字列テーブル、およびスプレッドシートファイル全体に適用するその他の情報を参照できます。

```
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/spreadsheetml/2006/main' xmlns:r='http://schemas.openxmlformats.org/officeDocument/2006/relationships'>")
        do stream.WriteLine("<sheets> <sheet name='\"_workSheetName_\"' sheetId='1' <span style='color:#2ecc71;'>r:id='rId1'</span>/>")
        do stream.WriteLine("</sheets> </workbook>")
    }
```

```
        do stream.%Save()
    }catch{
        set status=$$$N0
    }
    kill stream
    return status
}
```

rels/workbook.xml.rels

workbook.xmlパーツからの参照に一致するように、**rid1**というIDを持つリレーションを作成する必要があります。

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

    set isunix=$zcv($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/package/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.openxmlformats.org/officeDocument/2006/relationships/styles' Target='styles.xml' />")
    do stream.WriteLine("</Relationships>")
    try{
        do stream.%Save()
    }catch{
        set status=$$$N0
    }
    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/package/2006/relationships'>")
        do stream.WriteLine("<Relationship Id='rId1' Type='http://schemas.openxmlformats.org/officeDocument/2006/relationships/officeDocument' Target='workbook.xml'/>")
    do stream.WriteLine("</Relationships>")
    try{
        do stream.%Save()
    }catch{
        set status=$$$N0
    }
    kill stream
    return status
}
```

[ContentTypes].xml

静的ファイル（現時点では、ワークシートの数に応じた動的ファイル）はworkbookのワークシートとスタイルを紐づけます。Office Open

XMLファイルごとに、ZIPパッケージ使用されるコンテンツタイプを宣言する必要があります。

これは、[ContentTypes].xmlファイルで行います。

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

styles.xml

すべてのフォーマットがこのファイルに含まれます。現時点では、静的スタイルが追加されています（より動的なworkbook固有のスタイルに変換する予定です）。

Excelスタイル	ID	スタイル	Excelフォーマット
	1	デフォルト	テキスト
	2	#[Red]-#	数値
	3	####[Red]-###	数値
	4	yyyy/mm/dd	日付
	5	hh:mm	日付
	6	ヘッダーと中央揃え	テキスト
	7	ヘッダー2左寄せ	テキスト
	8	良い（緑ハイライト）	全般
	9	悪い（赤ハイライト）	全般
	10	どちらでもない（オレンジハイライト）	全般
	11	yyyy/mm/dd hh:mm	日付

```
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
readsheetsml/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/spreadsheetsml/2009/9/ac\"' xmlns:x16r2='\"http://schemas.microsoft.com/office/
spreadsheetsml/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

このファイルに日付が含まれます。シートの最初の行は列のタイトルです。
次の行には、最初の列にのみデータが含まれます。
デフォルトで列が自動調整されない場合は、ここで各列の列幅を定義します。

サンプル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>
```

サンプルExcel

A	B	
Name	Amount	
Jhon Smith	1000.74	
Tracy A	6001.74	

worksheet内の数式は、関数 タグを使って表現できます。

<c >

```
<f>B2*0.08</f >  
</c >  
<c >  
<f>B2+C2</f >  
</c>
```

そして最後にそれらをzip圧縮し、名前を.xlsxに変更します（unix zipを使用）。

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

Excelドキュメントを生成します。

以下のサンプルコードは、Excelドキュメントを生成します。

```
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のWriterクラスは、こちらの[xlsx.writer.xml.zip](#)にあります。

[#ObjectScript](#) [#Caché](#) [#InterSystems IRIS](#) [#InterSystems IRIS for Health](#)

ソースURL:

<https://jp.community.intersystems.com/post/cach%C3%A9-objectscript%E3%81%AB%E3%82%88%E3%82%8Bexcel%E3%83%89%E3%82%AD%E3%83%A5%E3%83%A1%E3%83%B3%E3%83%88%E3%81%AE%E7%94%9F%E6%88%90>