# The odsfile package:
# inserting opendocument spreadsheet as LaTeX tables[*]

Michal Hoftich (`michal.h21@gmail.com`)

July 23, 2012

## Contents

## 1 Introduction

This is LuaLaTeX package and lua library for reading opendocument spreadsheet (ods) documents from Open/Libre Office Calc and typeseting them as LaTeX tables. ods format consist of number of xml files packed in the zip file. This package uses LuaTeX's zip library and scripting to read xml content from this archive, which means that it is not possible to use this package with pdfLaTeX or XeLaTeX. On the other side, `odsfile.lua` library can be used from PlainTeX, ConTeXt or pure lua scripts.

Creation of this package was motivated by question[1] on site `http://tex.stackexchange.com/`. Development version of the package can be found at `https://github.com/michal-h21/odsfile`, all contributions and comments are welcome.

---

[*]Version 0.1, last revisited 2012-07-22.

[1]`http://tex.stackexchange.com/questions/60378/insert-libreoffice-table-as-input`

## 2 Usage

You can load `odsfile` classicaly with

`\usepackage{odsfile}`

There are two macros:

- `\includespread`

- `\tabletemplate`

\includespread  Main command is `\includespread`. It's syntax is:
`\includespread[⟨key-value list⟩]`
Options are:

**file** Filename of file to be loaded. You should specify this only on first use of `\includespread`.

**sheet** Name of sheet to be loaded. If it's not specified on first use of `\includespread`, then first sheet from the file is loaded. The sheet remains selected until another use of `sheet`.

| | |
|---|---|
| First | 2,2 |
| Second | 3,1 |

```
1 \begin{tabular}{l l}
2 \includespread[file=pokus.
    ods,sheet=List2]
3 \end{tabular}
```

**range** Selects range from table to be inserted. Range is specified in format similar to spreadsheet processors, like `a2:c4`, selecting cels starting at first column, second row and ending and third column, fourth row.

| | | |
|---|---|---|
| Hello | 1 | 3 |
| World | 2 | 4 |
| AA | 3 | 5 |

```
1 \begin{tabular}{lll}
2 \includespread[sheet=List1,
    range=a2:c4]
3 \end{tabular}
```

You can ommit some or both of the numbers:

| Label | Position | Count |
|---|---|---|
| Hello | 1 | 3 |
| World | 2 | 4 |
| AA | 3 | 5 |

```
1 \begin{tabular}{lll}
2 \includespread[range=a:c4]
3 \end{tabular}
```

| Label | Position |
|---|---|
| Hello | 1 |
| World | 2 |
| AA | 3 |
| BB | 4 |
| CC | 5 |

```
1 \begin{tabular}{ll}
2 \includespread[range=a:b]
3 \end{tabular}
```

<table>
<tr><td>1</td><td>3</td></tr>
<tr><td>2</td><td>4</td></tr>
<tr><td>3</td><td>5</td></tr>
<tr><td>4</td><td>6</td></tr>
<tr><td>5</td><td>7</td></tr>
</table>

```
1 \begin{tabular}{ll}
2 \includespread[range=b2:c]
3 \end{tabular}
```

**columns** Column heading specification. It can be either `head`, `top`, or comma separated list of values.

**top** Use as headers first line from the table.

| Position | Count |
| --- | --- |
| 2 | 4 |
| 3 | 5 |
| 4 | 6 |

```
1 \begin{tabular}{ll}
2 \includespread[range=b3:c
      5,columns=top]
3 \end{tabular}
```

Note that if you include whole table, first line is included twice:

| Label | Position | Count |
| --- | --- | --- |
| Label | Position | Count |
| Hello | 1 | 3 |
| World | 2 | 4 |
| AA | 3 | 5 |
| BB | 4 | 6 |
| CC | 5 | 7 |

```
1 \begin{tabular}{lll}
2 \includespread[columns=
      top]
3 \end{tabular}
```

in this case you can use

**head** use first row from selection as headings.

| Label | Position | Count |
| --- | --- | --- |
| Hello | 1 | 3 |
| World | 2 | 4 |

```
1 \begin{tabular}{lll}
2 \includespread[columns=
      head,range=a:c3]
3 \end{tabular}
```

**manualy specified list** If column headings are not specified in the file, you can set them manualy.

| title | amount |
| --- | --- |
| First | 2,2 |
| Second | 3,1 |

```
1 \begin{tabular}{ll}
2 \includespread[columns=
      head,columns={title,
      amount},sheet=List2]
3 \end{tabular}
```

**rowseparator** Rows are normally separated with newlines, if you really want, you can separate them with hlines

| Label | Position |
|-------|----------|
| Hello | 1 |
| World | 2 |
| AA | 3 |
| BB | 4 |

```
1 \begin{tabular}{lll}
2 \includespread[columns=top,
     sheet=List1,
     rowseparator=hline,
     range=a2:b5]
3 \end{tabular}
```

**template** Templates are simple mechanism to insert whole tabular environment with column specification. All columns are aligned to the left, if you want to do more advanced stuss with column specifications, you must enter them manualy as in all previous examples.

| Label | Position | Count |
|-------|----------|-------|
| World | 2 | 4 |
| AA | 3 | 5 |
| BB | 4 | 6 |
| CC | 5 | 7 |

```
1 \includespread[columns=top,
     template=booktabs,range
     =a3]
```

For more info about templates, see next section 3

# 3 Templates

If you don't want to specify tabular environment by hand, you can use simple templating mechanism to insert tabular environment by hand.

\tabletemplate          Templates are defined with macro
\tabletemplate{⟨*template name*⟩}{⟨*template code*⟩}
there is default template:

\tabletemplate{default}{-{colheading}-{rowsep}-{content}}

Code -{variable name} inserts one of the following variables:

**coltypes** This is code to be inserted in \begin{tabular}{coltypes}. In current version, it inserts l for left alignment column, for all columns of inserted table. It should be possible to use more intelligent method based on types of column content, or ods styles, maybe in future versions some of them will be used. If you want other alignment of columns now, you have to specify \begin{tabular}{column types} manually.

**colheading** Column headings.

**rowsep** It inserts row separator defined with rowsepartor key of \includespread. It is used in default style to delimit column headings and table contents.

**content** Tabular data.

**More powerfull template for booktabs package**

```
\tabletemplate{booktabs}{%
\\begin{tabular}{-{coltypes}}
\\toprule
-{colheading}
\\midrule
-{content}
\\\\ \\bottomrule
\\end{tabular}
}
```

Note use of the double \ in template definition – it is needed to pass them to the lua side.

# 4   Lua library

The lua library uses luazip library integrated to LuaTeX and LuaXML[2], pure lua library for working with XML files.

To use library in your code, you can use:

```
require("odsfile")
```

Function odsfile.load(filename) returns odsfile object, with loadContent() method, which returns lua table representing content.xml file. We can select sheet from the spreadsheet with odsfile.getTable(xmlobject,sheet_name). If we omit sheet_name, first sheet from spreadsheet is selected.

Data from sheet can be read with odsfile.tableValues(sheet, x1, y1, x2, y2). x1 - y2 are range to be selected, they can be nil, in which case whole rows and cells are selected. For converting of standard range expressions of form a1:b2 to this representation, function odsfile.getRange(range) can be used.

**Example usage – file** odsexample.lua

```
require "odsfile"

-- Helper function to print structure of the table
function printable(tb, level)
  level = level or 1
  local spaces = string.rep(' ', level*2)
  for k,v in pairs(tb) do
      if type(v) ~= "table" then
         print(spaces .. k..'='..v)
      else
         print(spaces .. k)
         level = level + 1
         printable(v, level)
```

---

[2]http://manoelcampos.com/files/LuaXML-0.0.0-lua5.1.tgz

```
      end
   end
end

local ods = odsfile.load("filename.ods")
local f, e = ods:loadContent()

-- Get First sheet from the table
body= odsfile.getTable(f)
-- Print structure of the range a4:b5
printable(odsfile.tableValues(body,odsfile.getRange("a4:b5")))
```

Run the example with `texlua odsexample.lua` from the command line, you shiuld get following result:

```
1
  1
    value=AA
    attr
      office:value-type=string
  2
    value=3
    attr
      office:value-type=float
      office:value=3
2
  1
    value=BB
    attr
      office:value-type=string
  2
    value=4
    attr
      office:value-type=float
      office:value=4
```

To convert this structure to LATEX tabular code, you can use following function:

```
function tableToTabular(values)
  local rowValues = function(row)
    local t={}
    for _,column in pairs(row) do table.insert(t,column.value) end
    return t
  end
  content = {}
  for i,row in pairs(values) do
    table.insert(content,table.concat(rowValues(row)," & "))
  end
  return table.concat(content,"\\\\\n")
```

```
end
-- Now use it with objects from previous example
print(tableToTabular(odsfile.tableValues(body)))
```

This example yelds

```
Label & Position & Count\\
Hello & 1 & 3\\
World & 2 & 4\\
AA & 3 & 5\\
BB & 4 & 6\\
CC & 5 & 7
```