

The odsfile package: accessing of the opendocument spreadsheet from L^AT_EX documents*

Michal Hoftich (michal.h21@gmail.com)

August 12, 2012

Contents

1	Introduction	1
2	Usage	2
3	Templates	5
4	Adding data	5
5	Loading and saving of the ods file	6
6	Lua library	7
7	Changes	9

1 Introduction

This is LuaL^AT_EX package and lua library for working with opendocument spreadsheet (ods) documents from Open/Libre Office Calc. Their contents can be read as L^AT_EX tables, can be pass to macros, you can also add new data to existing spreadsheets.

ods format consist of number of xml files packed in the zip file. This package uses LuaT_EX's zip library, LuaXML library¹ and lua scripting to read xml content from this archive, which means that it is not possible to use this package with pdfL^AT_EX or X_YL^AT_EX. On the other side, odsfile.lua library can be used from PlainT_EX, ConT_EXt or pure lua scripts.

*Version 0.2, last revisited 2012-08-12.

¹Pure lua library for working with xml files, it is available form CTAN or <https://github.com/michal-h21/LuaXML>

Creation of this package was motivated by question² 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.

2 Usage

You can load odsfile classically with

```
\usepackage{odsfile}
```

There are macros:

- `\includespread`
- `\tabletemplate`
- `\loadodsfile`
- `\savespreadsheet`
- `AddRow` environment

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 cells 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}
```

²<http://tex.stackexchange.com/questions/60378/insert-libreoffice-table-as-input>

You can omit some or both of the numbers:

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

```

1\begin{tabular}{l11}
2\includespread[range=a:c4]
3\end{tabular}

```

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

```

1\begin{tabular}{l11}
2\includespread[range=a:b]
3\end{tabular}

```

1	3
2	4
3	5
4	6
5	7

```

1\begin{tabular}{l11}
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}{l11}
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}{l111}
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}{l111}
2\includespread[columns=
head,range=a:c3]
3\end{tabular}

```

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

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.

Possible values:

tableline (default) Inserts `\\` character

hline Inserts `\\ \\hline`

newline Inserts blank line

user specified separator useful in conjunction with `rowtemplate` (p. 4), for example if you want to include sheets as plaintext or input for plotting functions.

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 stuff with column specifications, you must enter them manually 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

rowtemplate Enables to convert tabular data to something different than \LaTeX tables. Syntax for `rowtemplates` is similar to the table templates, variables are inserted with `-{number}`, where number is the position of the cell from beginning of the selection.

Hello:1, World:2, AA:3, BB:4, CC:5

```

\includespread[range=a2:b,
rowseparator={,\ },
rowtemplate={\textit
{--{1}}:--{2}}]

```

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 powerful template for the BOOKTABS package

```

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

```

4 Adding data

There is simple interface for adding new rows to the spreadsheet.

`AddRow` `AddRow[<row number>]` environment for adding new row to the current sheet.

Optional argument [*row number*] specifies where it should be inserted, if blank, it will be inserted at end.

Inside AddRow, you can use

- `\AddString{<text>}{<position>}`
- `\AddNumber{<number>}{<position>}`

Position specifies cell, where data should be added, if you leave it blank, it will be laced next to the previous one.

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

Label	Position	Count
Hello	1	3
third row		22
World	2	4
AA	3	5
BB	4	6
CC	5	7
last row		

```

1 \includespread[columns=head,
   template=booktabs]
2 \begin{AddRow}
3   \AddString{last row}{ }
4 \end{AddRow}
5 \begin{AddRow}[3]
6   \AddString{third row}{ }
7   \AddNumber{22}{3}
8 \end{AddRow}
9 \includespread[columns=head,
   template=booktabs]

```

5 Loading and saving of the ods file

`\loadodsfile`

You can explicitly load ods file with `\loadodsfile[<key val list>]{<filename>}`. This can be useful, if you only want to write some data to the file, otherwise it is better to use `\includespread`.

`\savespreadsheet`

For saving spreadsheets modified with AddRow, you can use `\savespreadsheet`. This command uses call to external zip utility, so you should have installed it and you have to call `luaLATEX` with `luaATEX --shell-escape filename`. `luaATEX` also must have write permissions for accessing the ods file. This command creates file `content.xml` in the current directory, so if external call fails, you can run

```
zip -r filename.ods content.xml
```

by hand.

6 Lua library

The lua library uses luazip library integrated to LuaTeX and LuaXML³, 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)
        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 should get following result:

³<https://github.com/michal-h21/LuaXML>

```

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(odsfiler.tableValues(body)))

```

This example yields

```

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

```


7 Changes

- v0.2**
- LuaXML is now distributed as separate library, so other projects can use it.
 - New AddRow environment for adding data to the ods file
 - New command \savespreadsheet for saving ods file
 - Bug fixes: corrected loading of the sheets, corrected behaviour of blank cell

v0.1 First version