XLSX Lua module

How to generate XLSX files in Lua

Version 1.0

1 Introduction

Generate human readable files from scripts is a difficult problem because most classical formats are not expressive enough or are too complicated to use. CSV files can not been formatted, text files are easy to generate but it is difficult to navigate inside and HTML files are powerful but difficult to share. XLSX files are very powerful to display and navigate into data, but it is difficult to generate.

MAQAO provides a Lua API to easily generate XLSX files. This document presents how its work, and how to use the API.

2 XLSX Classes

Several classes are implemented in the module and can be manipulated:

- XLSX_file class represents a regular xlsx file. It is considered as a set of sheets.
- XLSX_sheet class represents a single sheet. It contains cells or graphs
- XLSX_text class represents a cell and its text
- XLSX_table class is an abstract object representing a table. It is used to simplify the displaying process and it is needed to generate graphs.
- XLSX_graph class represents a graph.
- XLSX_comment class represents a comment linked to a cell

3 Basic Usage

This section describes main functions used for a basic usage of the API. Only a subset of module functions is presented in this section. To get all functions, go to the section 4. Exemples are described in section 5.

3.1 Create A File

The first step to create an XLSX file is to create an XLSX_file object using function `XLSX_file:create`. This function takes as parameter a string used as the name of the xlsx file to create. The .xlsx extension is added if it is missing in the given string. The function returns an initialized XLSX_file object.

3.2 Create A Sheet

Once the XLSX_file is initialized, some sheets can be added using the function `XLSX_file.add_sheet`. The function takes as parameters an existing XLSX_file object and the name of the sheet to create. The name does not contain spaces. It returns an initialized XLSX_sheet object.

3.3 Add Some Text

The API provides two ways to insert some text and other objects such as graphs or tables: a cursor increased when text is added and by giving location of the cell to update. Both ways can be used on the same sheet.

3.3.1Using Cursor

Using a cursor is the simplest way to insert text since positioning is intuitive. The cursor starts at cell “A1” or [1; 1]. When text is added, the cursor is moved on the column. When a new line is created, the cursor is set on the next row at the first column.

To add text, use the function `XLSX_sheet.add_text` with an existing XLSX_sheet object as first parameter and the text to display as the second parameter.

To go to the new line, use the function `XLSX_sheet.new_line`. The function takes as parameter an existing XLSX_sheet object.

3.3.2Using Location

Adding text using location is done using the function `XLSX_sheet.add_text`. The function takes at least three parameters: a XLSX_sheet object, the text to display (string or number) and a location. The location can be a single string representing coordinates formed with letters and numbers (for example “A5” or “BA16”) or a couple of integers greater than zero where the first one is the abscissa and the second one is the ordinate. If location is missing or wrong, the cursor is used instead. The function returns an initialized XLSX_text object representing the created cell.

3.4 Adding A Graph

The first step to display a graph is to create a XLSX_table containing data to display in the graph. To create the XLSX_table, use the function XLSX_table:new with two parameters, a string for the XLSX_table name and a 2D table containing all data. Each sub table represents a row; the first sub table and the first element of each sub table are names. An example is shown by figure 1. The function returns an initialized XLSX_table object.

Lua Code

```
local data = {
{"Experiment", "Time", "Count"},
{"Exp1", 245, 53},
{"Exp2", 263, 18},
{"Exp3", 598, 105},
}
```

Representation in the file

Experiment	Time	Count
Exp1	245	53
Exp2	263	18
Exp3	598	105

Corresponding Graph

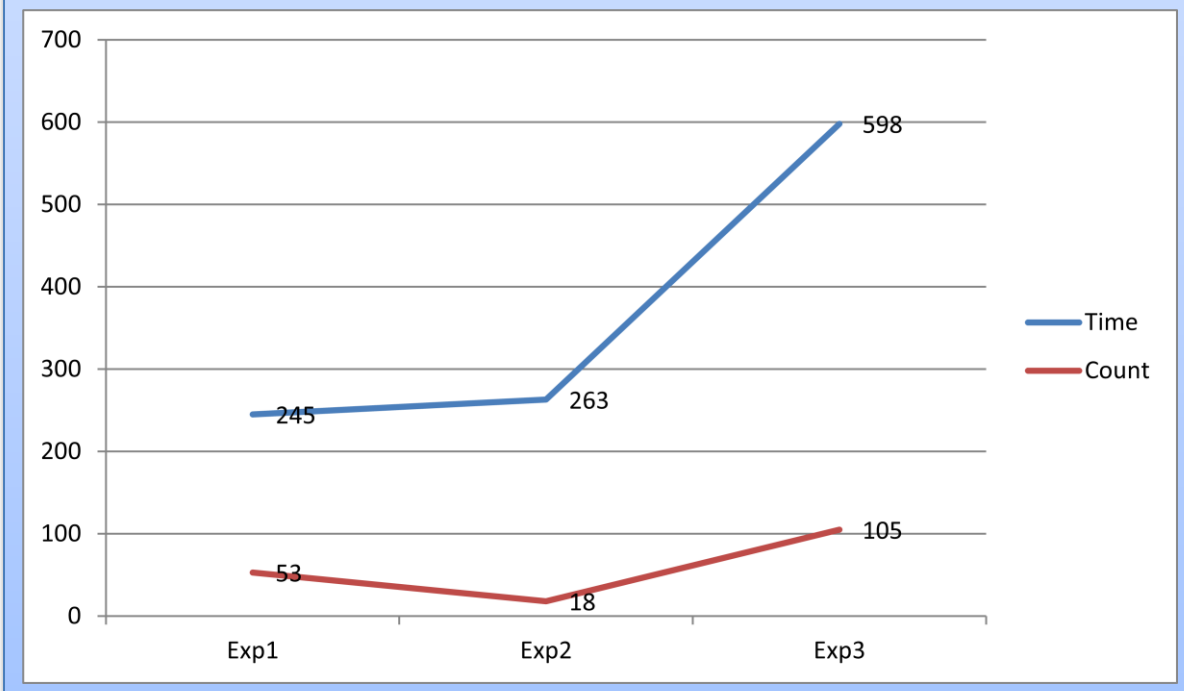

Figure 1: Structure of data to create a XLSX_table and a XLSX_graph

Once the XLSX_table has been created, it should be added in the sheet using the function `XLSX_sheet.add_table`. The first parameter is an existing XLSX_sheet, the XLSX_table is the second parameter and an optional location (either a string or a couple of integers) can be specified as third and fourth parameters.

Then a XLSX_graph object should be created using XLSX_graph:new function. The only needed parameter is the XLSX_table and a second optional parameter can be passed to specified the graph type (bar chart, line chart …). The function returns an initialized XLSX_graph object.

At least, to add the graph the function `XLSX_sheet.add_graph` must be used, with the existing XLSX_sheet object as first parameter, the XLSX_graph as second parameter and an optional location as third and fourth parameters.

3.5 Close The File

When everything is done, the file must be closed using the function `XLSX_file.save`. The function takes as parameter an initialized XLSX_file object. The function creates the .xlsx file.

4 Functions

This section describes all functions available in the XLSX module.

4.1 XLSX

```
XLSX:get_col_from_int (int)
```

Convert an integer into the corresponding column string. int begins at 1. Return a string.

Examples:
XLSX:get_col_from_int (1) = “A”
XLSX:get_col_from_int (2) = “B”

```
XLSX:get_int_from_col (str)
```

Convert a column string into the corresponding integer. Return an integer.

Example:
XLSX:get_int_from_col (“E”) = 5

```
XLSX:get_coordinates_from_text (str)
```

Convert a string into corresponding coordinates. Return a couple of integers.

Example:
XLSX:get_coordinates_from_text (“E2”) = (5, 2)

```
XLSX:get_text_from_coordinates (x, y)
```

Convert a couple of integer into a string representing a location.

Example:
XLSX:get_text_from_coordinates (5, 2) = “E2”

4.2 XLSX_file

```
XLSX_file:create (file_name)
```

Initialize an XLSX_file structure. file_name is a string representing the name of the xlsx file to create. The .xlsx extension is added if missing. The function returns the initialized XLSX_file or nil if there is a problem.

```
XLSX_file.add_sheet (file, sheet_name)
```

Add an empty sheet in an existing XLSX_file object. sheet_name is a string (which can be nil), representing the sheet name. If it is nil, a default name is set. The function returns the initialized sheet or nil if there is a problem.

```
XLSX_file.remove_sheet (file, sheet)
```

Remove a sheet from an existing XLSX_file. sheet can be either an integer or a string. If sheet is an integer, it is considered as the index of the sheet to remove. If sheet is a string, it is considered as the name of the sheet to remove.

```
XLSX_file.get_sheet (file, sheet)
```

Get an existing sheet from an existing file. sheet can be either a integer or a string. . If sheet is an integer, it is considered as the index of the sheet to return. If sheet is a string, it is considered as the name of the sheet to return. The function returns an existing XLSX_sheet or nil if no matching sheet is found.

```
XLSX_file.swap_sheets (file, sheet1, sheet2)
```

Swap position of two sheets in the sheet list. sheet1and sheet2 can be either integers or existing XLSX_sheet objects, but they must have the same type.

```
XLSX_file.save (file)
```

Create the .xlsx file corresponding to the given XLSX_file.

4.3 XLSX_sheet

```
XLSX_sheet.add_text (sheet, text, loc_x, loc_y)
```

Add some text in a new cell of an exisitng XLSX_sheet object. text parameter can be either a string, a number or an initialized XLSX_text object. If text is a string or a number, a new XLSX_text object is created, inserted into the sheet then returned by the function. I text is an existing XLSX_text object, it is inserted then returned. loc_x and loc_y are optional location parameters. As said, the function returns an initialized XLSX_text object.

```
XLSX_sheet.add_empty_cells (sheet, nb, loc_x, loc_y)
```

Add several empty cells in a existing XSLX_sheet. Mainly used when cursor is used. The second parameter is the number of empty cells to create, loc_x and loc_y are optional location parameters. The function returns the number of created cells.

```
XLSX_sheet.merge_cells (sheet, nb, text, loc_x, loc_y)
```

Merge several cells from a same row in an existing XLSX_sheet object. The second parameter is the number of cell to merge. text can be either a string or a number representing the text to display in the cell. loc_x and loc_y are optional location parameters. The function returns an initialized XLSX_text object representing the merged cell.

```
XLSX_sheet.add_hyperlinks (sheet, text, dst, loc_x, loc_y)
```

Add a hyperlink in an existing XLSX_sheet. text parameter is the text to display, dst is the target of the link, loc_x and loc_y are optional location parameters. dst can be either an existing XLSX_sheet belonging to the same XLSX_file than sheet to represent a link in the same xlsx file or a string to represent a link to another xlsx file. The string must have the following format: <file name>#<sheet name> where <file name> is the name of another xlsx file and <sheet name> is the name of a sheet inside <file name> xlsx file. The function returns an initialized XLSX_text object.

```
XLSX_sheet.new_line (sheet)
```

Move the cursor of an existing XLSX_sheet at the new line.

```
XLSX_sheet.get_cell (sheet, loc_x, loc_y)
```

Return an XLSX_text object located at given coordinates in an existing XLSX_sheet object. loc_x and loc_y are optional location parameters. If there is no given location, the function returns the cell at cursor current location, else the function returns the cell at given location. If no cell exists at given location, an empty cell is created and returned.

```
XLSX_sheet.get_location (sheet)
```

Get current location (cursor coordinates) in a given XLSX_sheet

```
XLSX_sheet.add_table (sheet, table, loc_x, loc_y)
```

Add an existing XLSX_table object into a XLSX_sheet object. loc_x and loc_y are optional location parameters. If no location is given, the cursor location is used.

```
XLSX_sheet.add_graph (sheet, graph, loc_x, loc_y)
```

Add an existing XLSX_graph object into a XLSX_sheet object. loc_x and loc_y are optional location parameters. If no location is given, the cursor location is used.

```
XLSX_sheet.set_columns_size (sheet, start, stop, size)
```

Set some sheet columns size (in cm). start and stop are columns index (integers greater than 0), all columns between start and stop are resized.

```
XLSX_sheet.set_rows_size (sheet, start, stop, size)
```

Set some sheet rows size (in cm). start and stop are rows index (integers greater than 0), all rows between start and stop are resized.

```
XLSX_sheet.add_comment (sheet, text, loc_x, loc_y)
```

Add a comment on an existing XLSX_sheet object. The second parameter is a string with the comment text, third and fourth parameters are used for location.

If location is nil, the comment is added on the current cursor location (last modified cell).

4.4 XLSX_text

```
XLSX_text:new (value)
```

Initialize a XLSX_text object with a given value (string or number). The object is not inserted in any sheet.

```
XLSX_text.set_bold (text, bool)
```

Set bold (bool == true) or not (bool == false, default value) the text of an existing XLSX_text object.

```
XLSX_text.set_underline (text, bool)
```

Set underlined (bool == true) or not (bool == false, default value) the text of an existing XLSX_text object.

```
XLSX_text.set_strike (text, bool)
```

Set stroke (bool == true) or not (bool == false, default value) the text of an existing XLSX_text object.

```
XLSX_text.set_italic (text, bool)
```

Set in italic (bool == true) or not (bool == false, default value) the text of an existing XLSX_text object.

```
XLSX_text.set_size (text, size)
```

Set the font size of an existing XLSX_text object. size should be a number or a string representing the font size.

```
XLSX_text.set_color (text, color)
```

Set the font color of an existing XLSX_text object. Color must be a string representing a hexadecimal value in RGB mode (00rrggbb). Some default colors are defined in file XLSX_consts.lua, in XLSX.consts.colors table.

```
XLSX_text.set_background (text, color)
```

Set the background color of an existing XLSX_text object. Color must be a

string representing a hexadecimal value in RGB mode (00rrggbb). Some default colors are defined in file XLSX_consts.lua, in XLSX.consts.colors table.

```
XLSX_text.set_font (text, font)
```

Set the font of an existing XLSX_text object. font must be a string with the font name.

```
XLSX_text.set_alignment (text, alignment)
```

Set the font alignment on an existing XLSX_text object. alignment must be an element of XLSX.consts.align table.

```
XLSX_text.set_border (text, location, type)
```

Set borders on an existing XLSX_text object. Both location and type are elements of XLSX.consts.borders table. location represents where borders should be set and type represent the type of border to use. Borders are cumulative. This means that two calls of this function with different locations set two borders on the cell.

4.5 XLSX_table

```
XLSX_table:new (name, data, orientation)
```

Create a new XLSX_table object. name is the table name, data is a 2D table describe in next paragraph and orientation is an optional constant in {XLSX.consts.orientations.columns, XLSX.consts.orientations.lines}. Default orientation value is XLSX.consts.orientations.columns.

data is a 2D table where first row and first element in each other row is a name. Other elements can be string or number. According to the orientation constant, the table will be displayed differently, such as shown by figure 2. Orientation does not have any effect on graph rendering.

Lua Code

```
local data = {
{"Experiment", "Time", "Count"},
{"Exp1", 245, 53},
{"Exp2", 263, 18},
{"Exp3", 598, 105},
}
```

Orientation == XLSX.consts.orientations.columns

Experiment	Time	Count
Exp1	245	53
Exp2	263	18
Exp3	598	105

Orientation == XLSX.consts.orientations.lines

Experiment	Exp1	Exp2	Exp3
Time	245	263	598
Count	53	18	105

Figure 2: Effects of orientation for XLSX_table objects

4.6 XLSX_graph

```
XLSX_graph:new (table, graph_type)
```

Create a new XLSX_graph object. table is an existing XLSX_table (not already used by another graph) corresponding to graph data. graph_type is a constant defined in XLSX.consts.graph_types table.

```
XLSX_graph.set_height (graph, nb_rows)
```

Set the graph height. graph is an existing XLSX_graph object and nb_rows is the new size (in number of rows) of the graph.

```
XLSX_graph.set_width (graph, nb_columns)
```

Set the graph width. graph is an existing XLSX_graph object and nb_columns is the new size (in number of columns) of the graph.

4.7 XLSX_comment

```
XLSX_comment.set_color (comment, color)
```

Set the comment background color. comment is an existing XLSX_comment object and color is a string representing the color with the format “#rrggbb” (warning, it is not “00rrggbb” such as other colors).

```
XLSX_comment.set_font_color (comment, color)
```

Set the comment font color. comment is an existing XLSX_comment object and color is a string representing the color with the format “00rrggbb”.

```
XLSX_comment.set_width (comment, width)
```

Set the comment width. comment is an existing XLSX_comment object and color is a number representing the new comment width in px.

5 Examples

An example script called tutorial_XLSX.lua is available in the same directory. It can be run with the command

```
$ maqao tutorial_XLSX.lua
```

It produces a file called XLSX_tutorial.xlsx in the current directory.

6 Known Bugs

- If some style (background, borders, color font …) are set on the first cell of a sheet (cell A1, initial location of the cursor), it could be used as default style for the whole sheet. A way to avoid this is to move the cursor on the next line using `XLSX sheet.new line` or to use location to avoid the first cell.
- Graph rendering is bad on Open Office, but good on Microsoft Excel.