New column names in an Excel document

If you want to export data from your FileMaker solution regularly as Excel document, perhaps you want to change the column headers every time. Maybe because you have a long grown database and the column names do not fit anymore. Maybe you work in an international company and need the column names in different languages. Perhaps you changed the column names manually so far, but now is the time to automate it. In huge databases this can be a very annoying task and waste a lot of time. I want to show you one way to make this automatically for your database with the help of the MBS FileMaker Plugin and LibXL.

In this example we load the previously exported Excel file, change the column names and save it with the optional saving dialog.

At first we initialize LibXL. We add the Script InitXL, which you can find in the example databases coming with the plugin, to our solution and call that script, if LibXl is not already initialized.

```
If [ MBS("XL.IsInitialized") ≠ 1 ]
        Perform Script [ Specified: From list ; “InitXL” ; Parameter: ]
End If
```

If there is no file in the container, we call the open dialog and store the file in a container field. We load it up to the memory via XL.LoadBook function and save the reference number in the variable $book. Most XL functions need this reference number to work on the particular excel document and you can have several of them in memory at the same time. We check with IsError function whether there was an error.

```
If [ IsEmpty(Excel test::Input) ]
        Insert File [ Excel test::Input ]
End If

Set Variable [ $book ; Value: MBS("XL.LoadBook";Excel test::Input) ]
If [ MBS("IsError") ]
        Show Custom Dialog [ "Error loading the Excel docume…" ; $book ]
        Exit Script [ Text Result:]
End If
```

For the next steps, the changing of the column names, we need the knowledge of the Excel document structure. We must know the order of the column names and which one of them we want to change and the name of the working sheet which we want to change.

In our example the working sheet name is „contacts“ and we have the column names: abc_Firstname, last name, Company_Phone and company. We want to change abc_Firstname to first name and Company_Phone to phone number. The column indexes start with index 0. We change column name 0 and 2, accordingly.

```
Set Variable [ $sheet ; Value: "contacts" ]
Set Variable [ $row ; Value: 0 ]
Set Variable [ $r ; Value: MBS( "XL.Sheet.CellWriteText"; $book; $sheet; $row; 0; "test") ]
```

In a variable $sheet we save the name of the working sheet and in $row the number of the line which should be changed. In the test version of LibXL the fist line is not editable, this is the reason, why our example file have have a blank row to the beginning of the document. Otherwise the load of the file would override the first row in test version. To determine whether the LibXL version is a test version or a full version license you try to write in the first row(0), if it succeed we have a full version and have to rewrite the field content. We want to change the cell of row 0 and column 0, because of this we do not need to rewrite the cell content back.

```
If [ MBS("IsError") ]
        Set Variable [ $row ; Value: $row + 1 ]
End If
```

If this function have an Error we knew it is using a test version and increment $row. Now $row have the value 1. You can remove the block to test this once you bought the license for LibXL.

```
Set Variable [ $r ; Value: MBS( "XL.Sheet.CellWriteText"; $book; $sheet; $row; 0; "first name") ]
Set Variable [ $r ; Value: MBS( "XL.Sheet.CellWriteText"; $book; $sheet; $row; 2; "phone number") ]
```

The writing command have the following structure:

MBS("XL.Sheet.CellWriteText"; reference number; work sheet; row number; column number; new Text)

The work sheet can be given by zero based index or by the name. If name is given, the plugin will check which sheet has the given name. Row and column numbers are zero based. If needed, the XL.Sheet.AddrToRowCol function can help to convert from usual references like C10 to the numbers like row = 9 and column = 2.

After the changing we want to save $book in an Excel file.

```
Set Variable [ $data; Value: MBS("XL.Book.Save"; $book; "newColumnName.xlsx") ]
If [ MBS("IsError") ]
        Show Custom Dialog [ "Save failed" ; $data ]
Else
        Set Field [ Excel test::Output ; $data ]
End If
Export Field Contents [ Excel test::Output ; Create directories: On ]
```

The XL.Book.Save function saves the document and returns a container value. For parameters the file name extension is imported. We use .xlsx here for input and output. We store our Excel file in the variable $data and can now decide what we want to do with the Excel document. In our example we put the document to a field and export it with an optional saving dialog to an external file.

At last we must free the memory used for the document by passing the reference number to XL.Book.Release function. This also frees all sheets, formats and fonts for that document.

Set Variable [$r ; Value: MBS("XL.Book.Release"; $book)]

Licenses of the MBS FileMaker Plugin and LibXL are available on the Monkeybread Software website. Both you can be tested before purchase with a trial license.

Have a lot of fun with this tutorial!
If you have questions, please do not hesitate to contact me.

Written by Stefanie, edited by Christian.