



Flexcel Reports Developers Guide

Documentation: May, 2010
Copyright © 2010 by tmssoftware.com bvba
Web: <http://www.tmssoftware.com>
Email : info@tmssoftware.com

Table of contents

- Introduction..... 3
- About Excel Reporting 4
- Organization of a FlexCel Report 5
 - Data Layer 5
 - Interface Layer 5
 - Presentation Layer 5
- Data Layer 6
- Interface Layer 7

Introduction

FlexCelReport is a component designed to create Excel files by modifying a template.

About Excel Reporting

FlexCel gives you two ways to create an Excel file, with FlexCellImport (API) or with FlexCelReport (Template).

Each method has its good and bad things, and it is good to know the advantages and drawbacks of each.

Creating a report using FlexCellImport is a low level approach; we might if you want compare it with programming on assembler. Assembler programs are fast and can use all the power on your machine. Also, assembler programs are difficult to maintain and difficult to understand to anyone who was not the author. And sometimes to the author too.

Similar to assembler, creating a report with FlexCellImport will result on a really fast report, and you have access to the entire API, so you can do whatever you can do with FlexCelReport and more. After all, FlexCelReport uses FlexCellImport internally.

And, again similar to assembler, FlexCellImport reports can be a nightmare to maintain. When you reach enough number of lines like:

```
FlexCellImport.CellValue[3,4] := 'Title'; Xlsformat.Font.bold:=true;  
Xlsformat.Font.Size:=14;  
XF := FlexCellImport.AddFormat(xlsformat);  
FlexCellImport.SetCellFotmat(3,4, XF);
```

Changing the report can become quite difficult. Imagine the user wants to insert a column with the expenses (And don't ask me why, users always want to insert a column). Now you should change the line

FlexCellImport..SetCellFotmat(3,4, XF); to FlexCellImport.SetCellFotmat(3,5, XF);

But wait! You need to change also all references to column 5 to 6, from 6 to 7... If the report is complex enough, (for example you have a master detail) this will be no fun at all.

But there is something much worse on using FlexCellImport directly. And this is that again, similar to assembler, only the author can change the report. If your user wants to change the logo to the new one, he needs to call you, and you need to recompile the application and send a new executable.

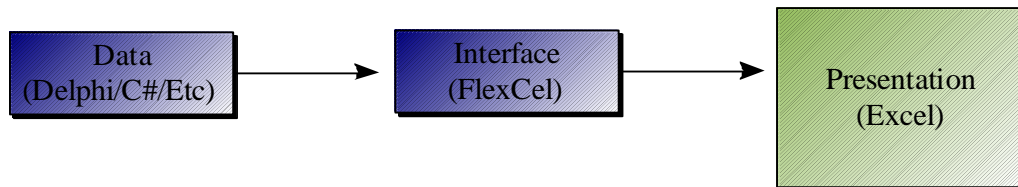
FlexCelReport is a higher level approach. The design is cleanly separated on three different layers, data layer, interface layer and presentation layer, and most of the work is done on the presentation layer, with Excel. You design the report visually on Excel, and you mess as little as possible with the data layer. If the user wants to change the logo, he can just open the template and change the logo. If he wants to insert a column, he can just open the template and insert a column. And the application does not even need to be recompiled.

As with any higher level approach, it is slower than using the API directly, and there are some things where it is more limited. But all in all, reports are really fast too and the things you cannot do are not probably worth doing anyway.

So, the option is yours. For normal reports we highly recommend that you use FlexCelReport. Even when it can take a little more time at the beginning that just hurry and start coding, it will pay on the long run. And much.

Organization of a FlexCel Report

A FlexCel report can be seen as three different modules working together to create the Excel file. Different from a “Coded” report where you mix data access with presentation, here each part has its own place, and can be developed separately by different people with different skills.



Data Layer

This is the layer where we read our data from a database or memory and prepare it to be read by the interface layer.

Interface Layer

This layer's mission is to “answer” petitions from the Presentation layer, providing the data it needs to create a report.

Presentation Layer

This is the most complex and changing layer of the three. Here is where you design all the visual aspects of the report, like data position, fonts, colors, etc.

The big advantage of this “layering” is that they are somehow independent, so you can work on them at the same time. Also, Data and Interface layers are small and do not change much on the lifetime of the application. Presentation does change a lot, but it is done completely on Excel, so there is no need to recompile the application each time a cell color or a position changes.

Remember: The data flow goes from the Presentation layer to the Data layer and back. It is the presentation that asks FlexCel for the data it needs, and FlexCel that in turn asks the data. It is not the application that tells FlexCel the data it needs (As in SetCell(xx)), but FlexCel that will ask for the data to the application when it needs it.

On this document we are going to speak about Data and Interface layers. The Presentation layer is discussed on a different document, “End User Guide” because it is complex enough to deserve it, and because it might be changed by someone who only understands Excel, not Delphi. And we don't want to force him to read this document too.

Data Layer

The objective of the data Layer is to have all data ready for the report when it needs it. Currently, there are four ways to provide the data:

1. Via Report Variables. Report variables are added to FlexCel by setting the “FlexCelReport.Value” property. You can define as many as you want, and they are useful for passing constant data to the report, like the date, etc. On the presentation side you will write `#.ReportVarName#`. each time you want to access a report variable.
2. Via User defined functions: User defined functions are functions you define on your code that allow for specific needs on the presentation layer that it can't handle alone. For example, you might define a user function named “NumberToString(number: integer) that will return “One” when number=1, “Two” when number =2 and so on. On the presentation layer you would write `#.NumberToString#.2` to obtain the string “Two”
3. Via DataSets: You can use any TDataSet object in the Datamodule or form where the FlexCelReport is located as a datasource for your report. In the template, you will write “##dataset##field” DataSets are the primary way to provide data to FlexCel.
4. Via TFlxMemTable: You will normally use TFlxMemTable when you have data that is not into a dataset but has many rows of records, so using report variables is not a good option (for example in a TList, or in your own business object collection). A TFlxMemTable can work in two ways: In normal mode it will work as a simple memory table, where you will copy the data from your own objects. This is the simplest mode, but it holds the objects twice in memory, once in your original objects, and once in the TFlxMemDataSet. So you can use a TFlxMemTable also in virtual mode where you feed the data with events. In this mode you won't have to hold the data twice, so it is the best option for large datasets.

Interface Layer

This is the simplest of the three, as all work is done internally. To setup the interface layer, you only need to tell FlexCel which datasets, values and user functions it has available from the data layer. For datasets, you just need to drop them in the Datamodule where FlexCelReport is located (or the datamodule where FlexCelReport points to). For values, set the “Value” property in FlexCelReport. For User Defined functions, create published methods in the datamodule FlexCel points to. Once you have told FlexCel what it can use, just call FlexCelReport.Run and this is all.

Note: As simple as it is, it is worth spending some time thinking what tables, functions and variables from the data layer you want to make available for the final user. Limiting it too much might mean cutting the power of the user to customize his report. And just adding everything might have some security issues, as the user might get access to tables he is not expected to.