tmssoftware.com

FlexCel Studio 5

INTRODUCTION

If you develop applications for the .NET Framework, be it Winforms, ASP.NET or WPF, you are likely to need to interface with Excel® sooner or later. You might need to create Excel files that your users can analyze, or import Excel files created by your users into your application.

Even when this is a very common task, neither Excel nor the .NET framework provide a good way to create this interface. Among the usual methods we can find:

1) Using OLE Automation: Besides being slow and not reliable, requires you to have an Excel license for each installation of your application, even if it is ASP.NET running in a server.
2) Using an OLEDB database driver for Excel: While better than OLE automation in regard to speed or stability, it has very limited formatting support.
3) Saving as CSV and importing the file in Excel: Not only it has limited format support, but it also might have problems with locales, in places where the decimal separator is the "," and not the"."
4) Excel 2007 provides a new xml format with the idea of overcoming those limitations, but creating a full featured xslx file in xml can be as complex if not more than creating a binary xls file. Just because "it is xml" it does not mean it is easy, and in fact, xlsx and xls file formats aren't that different. One format stores the data in binary and the other in text, but complexity is similar and data structures too.

FlexCel Studio is designed to overcome these problems. On a very affordable package, you get a dll that is completely self-contained and that allows your application to read and write Excel files in a fast and reliable way, without needing to install Excel or any other external file. You can forget about all of the headaches associated with the above solutions and just get your job done. Better and faster than what you would with any of the other mentioned solutions.

And FlexCel is much more than just an API to read or write xls files. On a high level view, FlexCel contains:

1) **Reading/writing API for xls/xlsx files** that you can use to manipulate Excel files. Different from other third party alternatives, FlexCel API is specially designed to **modify** documents, and this means it can modify xls files that it can't completely parse. For example, if you have a file with macros, even when FlexCel does not currently offers the ability to read or write macros, it will preserve them. Virtually every feature of the original Excel file you are modifying is preserved, so you do not have to worry about losing things when editing files. Note that FlexCel modifies xls files, xlsx need to be parsed.
2) **A recalculation engine**. Very accurate and fast, with more than 200 functions supported, it can recalculate almost any formula you put in your document (including array formulas or linked formulas to other documents). This way, you can run FlexCel on an environment where Excel is not installed (like a server) and create pdf files with the actual calculated data.
3) **A reporting engine**. On the foundation of the xls/xlsx API mentioned earlier, there is a full reporting engine implemented. It works like a reporting tool, but using Excel as the report designer, and allowing any power user to modify the reports in Excel without changing the code. You design the entire report directly in Excel instead of code, providing a clean separation between data and presentation that will make the job of modifying those reports much easier later.
4) **A state-of the-art rendering engine**. Also included with FlexCel is a rendering package that can natively print and preview an Excel file without having Excel installed, **export the file to PDF without having Excel or Acrobat** or any third party add on, or export the file to images. All the work is done on pure C# code, and the results are very detailed. From conditional formats to 2-D charts, from rotated text to page headers, almost every Excel feature is rendered, producing amazingly similar documents to the ones you would get printing directly from Excel. And since FlexCel uses GDI+ to do the rendering, printing is device independent, meaning that it will print the same everywhere. While printing with Excel can result in very different things depending on the printer, FlexCel output is virtually the same everywhere you print.
5) **An HTML exporting engine**. As with every part of the FlexCel framework, a lot of attention has been dedicated to the detail. While HTML export is not as exact as the PDF export, due to limitations on the HTML standard, output is very similar too. And not only output is great; it is also fully standards compliant and compatible with all modern browsers.
6) **A basic PDF API** to create PDF files. In order to natively convert Excel files to PDF, we had to develop our own PDF library. Even when it is not fully featured and its main goal is to be used for the conversion, it is available to you and you can use it to create non complex standalone pdf files,

Mentioning all the features available on FlexCel on this small paper would be impossible, even when we will try to resume the most important things on the following lines. But the main idea of this paper is to get you started so you download the trial and test it for yourself, because you might find a lot of more things than what is mentioned here.

POWERFUL

If you have been working some time with Excel, you know it is a quite complex application, with thousands of options and documented and undocumented features. While we do not support every single feature on Excel, we support a huge range of them. And as FlexCel strength is on modifying files, it can support things it does not actually understand. For example, there is no built in support to create a pivot table. But you can create an empty pivot table on Excel, open the file with FlexCel, insert rows with the data and save the new file. The new file will contain a completely valid Pivot Table with the correct data. (You can see an actual example of this on the "Pivot Table" demo)

There are just too many features accumulated over the years to be mentioned here. From the very accurate recalculation that honors things like 1904 dates or "precision as displayed" to the ability to read and write encrypted files. From the ability to read and write pxl (pocket Excel) files to the chart and conditional format rendering, to the ability to read and save from/to streams without creating temporary files.

As they say a picture is worth a thousand words, let's take a look at the following one:

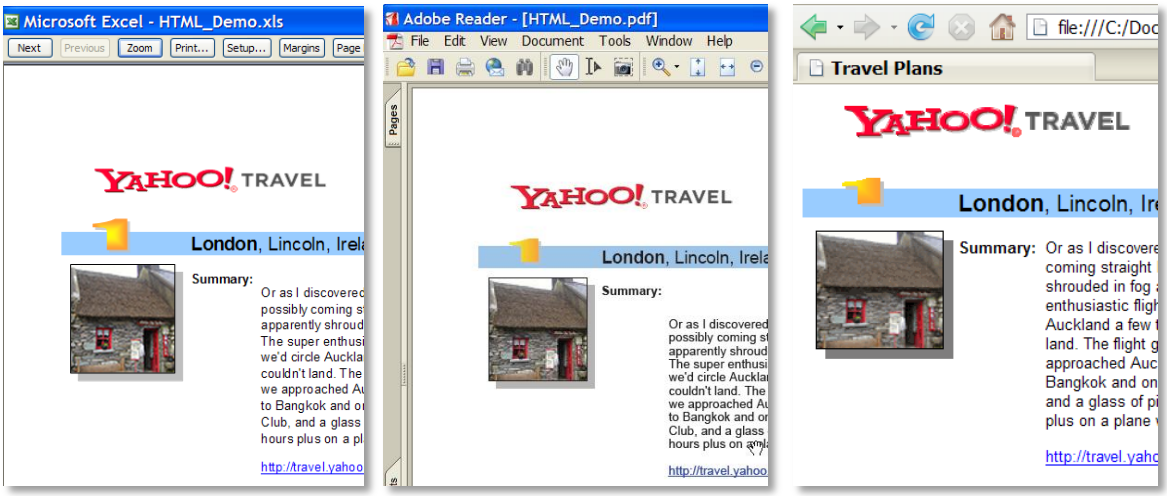

Image 1: An Excel file generated with FlexCel and the same file natively exported to PDF and HTML

It shows a file created with FlexCel (You can give it a try yourself with the HTML Demo on the FlexCel package) and the same file exported to PDF and HTML with FlexCel. The first thing you realize looking at the xls file to the left is that this is not a CSV export. There is rich format, some Wordart (the number "1"), a static image, a dynamic image with a shadow and one hyperlink. And as you can see at the pdf and html files at the right, everything has been exported, including the hyperlink (that is fully functional on the pdf and html files too). The pdf and html output is not exactly the same as the Excel one, but we think it is really close and one of the best you will get. And we are always working to make it better.

But we could be speaking for ages about all the available things. As this is not the main idea of this document, we will refer you to http://www.tmssoftware.com/flexcel/featuresflexcel.htm to seem more of them. And as in so many other places in this document, we would like to

encourage you to download the more than forty included demos and see all the features for yourself.

FAST

FlexCel for .NET is fast. Really fast.

We are always profiling the code and evaluating the algorithms to make FlexCel run as fast as they it can possibly run, and no more. Correctness is always first, as it should be, but performance is an important second.

But take note, you will not find any silly benchmarks on our page measuring the thousands of cells you can write on a millisecond, even when we know FlexCel has very good numbers on any benchmark. But we just look at them as meaningless, because they measure a very specific aspect of very complex application, and on real life you mix a lot of things together. It doesn't matter if you over optimized a specific method so it takes four nanoseconds instead of five, if when you try to do some real work the component starts to crawl. Those simple benchmarks are not answering the real questions, which are more like: How does this component scale when reading or writing tens of thousands of cells, with images, charts and formatting? How does it handle many components working on parallel threads?

We do not optimize FlexCel for benchmarks but for real use, with real data that our users and ourselves provide, everything working together as it will on the real life. With shared data for shared threads. Trying to keep memory usage as low as possible (considering that the full file is in memory) so the component does not start to swap to the hard disk. Trying new and more efficient algorithms to do the same things. Because performance is not a goal that you achieve and forget; it is a path that you are always traveling without actually getting to the destination.

We will not try to tell you that large and complex reports will not take their time to complete. They will, because creating a complex Excel file is a memory and computationally expensive task. But we make our best so complex files degrade performance as little as possible, and you still get reasonable times. And please do not take our word for it. Just download the FlexCel trial and verify it by yourself, we are pretty sure you will not be disappointed.

SOLID

Having the fastest and more powerful component would be meaningless if it crashes every third document it opens. We have many years of experience building Excel components, and several thousand regression tests that have been accumulated over that time that ensure FlexCel "just works". It might not be the sexiest of the features, or something that you even realize, but the fact that it works in a reliable and expected way is the most important thing to us. There is nothing worse than having your project freeze because of a bug on a third party code, and we work very hard to avoid this situation as much as we possibly can.

PORTABLE

FlexCel doesn't have any p/invokes on its code, and the most of it is plain managed code. On Version 4 we introduced some code that needs unmanaged permissions, but it is only used to convert images to black and white or 256 colors. This means that if you are not using the color-conversion capabilities of the component to export xls files to images, FlexCel can run on an environment without unmanaged permissions, like medium trust (for example when deploying to shared hosting).

And the fact that we do not use p/invokes also means that FlexCel works on **Mono**, on **Compact Framework**, on **64 bits**, or wherever the .NET framework goes. Had we made just one call to a Win32 dll, all code would be tied to the Win32 platform.

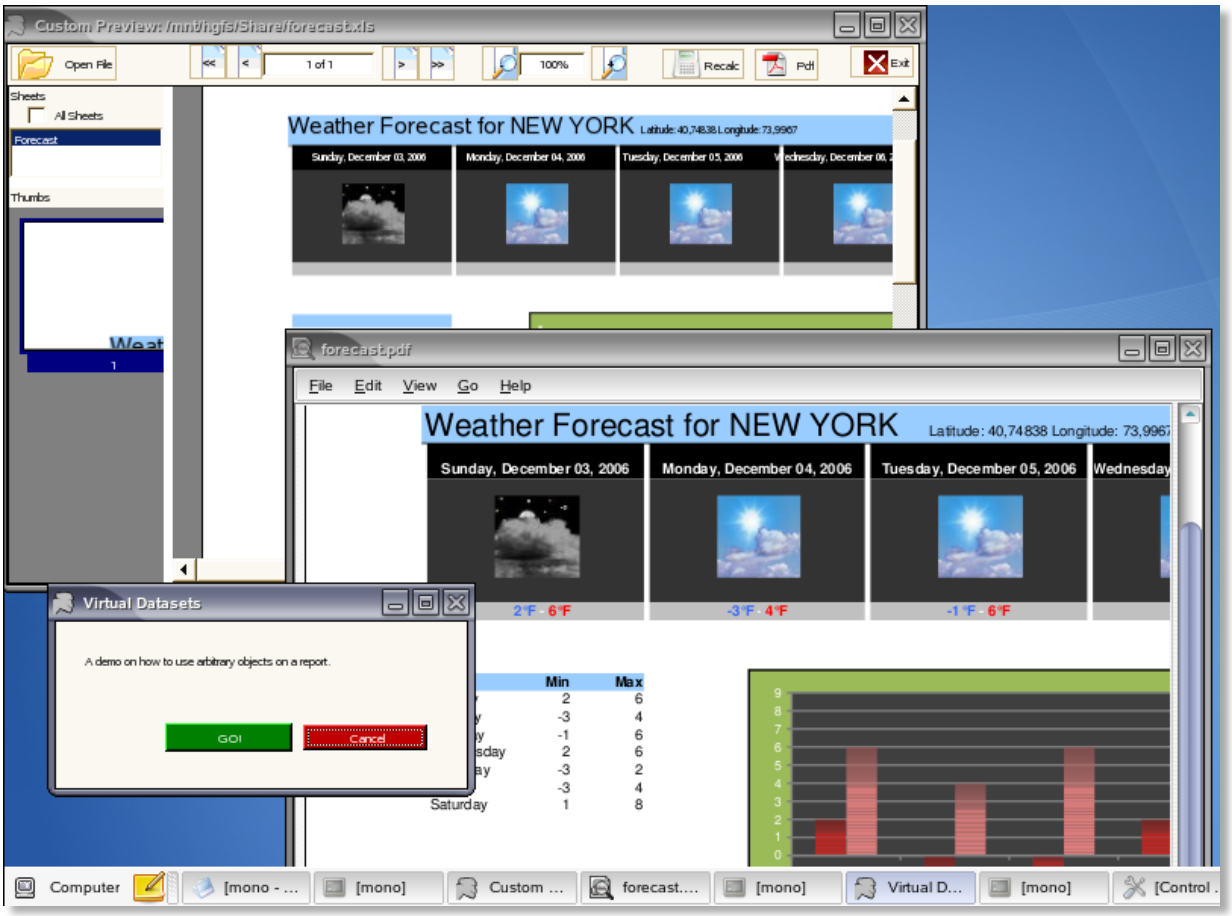

Image 2: Screenshot of the "CustomPreview" and "Virtual DataSets" demos running under Mono

As you can see from the screenshot, you can even run the FlexCel previewer under Mono. Mono support for Winforms is not near perfect yet, so there are errors on the rendering and you could not use the previewer on a real application. But for server side work (for example to create the pdf you can see on the second window, or to create any xls or html file) FlexCel under Mono just works, and the final result is the same as you would have got under Windows.

DOCUMENTED AND SUPPORTED

FlexCel documentation is divided on three different layers:

1) **Reference**: We provide a complete API documentation where every method is documented.
2) **Concepts**: Because reference is not enough to get started, we also include conceptual pdf files where we explain the theory underlying the tasks.
3) **How To**: More than 40 demos on how to perform the things you want to do, each one documented on its own. All of them are accessible from a "MainDemo" demo browser so you can find them easily, or they can be run as standalone applications when you want to study them on their own.

And as expected, we also provide email and newsgroup support for those times where the documentation is just not enough.

AFFORDABLE

We believe you should not have to sell a house to buy our components, nor to hire a lawyer to understand our licensing terms. So we make it simple:

One license per developer and this is it. Each license is royalty free, and never expires (you might need to pay for upgrades in the future, but you can continue using the code you bought forever). This means that if you are the only developer working on a project, the only thing you need to buy is a Single Developer License. No server licenses, no usage fees. If you are part of a team with four or more developers, you can buy a Server License that allows unlimited developers on unlimited projects to use FlexCel, for a very small price.

If you had developed your application using OLE Automation and you needed to install Excel in just one machine that didn't had it (because it was a server or they were running Open Office), you would probably spend more money than to buy an unlimited royalty-free license for FlexCel.

FULL SOURCE CODE

The money you spend buying a third party component is just a small fraction of the money you are going to spend using it. Once you buy a component, you will spend hundreds of hours coding over it, and this is a much bigger and somehow hidden price than the original money you paid for the component.

So, what happens if someday you can't get the component anymore? If you need to recompile it on a newer framework version, and you have a binary dll, you are out of luck. We believe you have the right to have the full source code, so you can know that whatever happens to us, you can still compile your application. And you can have a peek inside to know why something is not working as you hoped it would. So all FlexCel licenses come with FULL source code. Because you shouldn't buy components that do not come with it.