Volume

5

FLEXCEL STUDIO FOR VCL AND FIREMONKEY

TMS Software



Exporting Excel files to HTML

Table of Contents

TABLE OF CONTENTS ..... **I**
INTRODUCTION **1**
CREATING HTML FILES **1**
INTRODUCTION: 1
CREATING A FILE USING FLEXCELHTMLEXPORT: 1
Naming the created files 2
Embeding the generated HTML inside your own pages 2
CUSTOMIZING THE GENERATED HTML **3**
GENERAL CUSTOMIZATION 3
CUSTOMIZING THE RAW HTML 3
CUSTOMIZING THE SHEET SELECTOR 3
CUSTOMIZING THE FONTS 4
EXPORT TO HTML AND FLEXCEL RECALCULATION **5**
USING RELATIVE HYPERLINKS **5**

Introduction

One of the things FlexCel allows you to do is to export your Excel files to HTML. Note that as always, we do our best to create the HTML files as closely as possible to the original Excel file, but there are some restrictions in HTML that make it not as good as a PDF export. HTML is a "Flow" format, opposed to PDF (which is a "Fixed Page" format), and this means the browser can resize and reposition elements depending on the output medium. With this limitation in mind, we will study now how this is done.

Creating HTML files

Introduction:

You can create an html by using the **FlexCelHtmlExport** component.

Creating a file using FlexCelHtmlExport:

FlexCelHtmlExport is a straightforward component. You would normally set its **Workbook** property to the xls file you want to export, and then call **Export**, **ExportAllVisibleSheetsAsTabs** or **ExportAllVisibleSheetsAsOneHtmlFile** depending on the type of export you want to do. As always, we will not cover those methods or the properties on FlexCelHtmlExport in detail, since they are described in the reference and it makes no sense to repeat the information here. This document is about the conceptual part of creating HTML files.

Naming the created files

The first thing we need to note is that creating an HTML file is different from creating a PDF or other types of files in the sense that you actually create many files for a single xls/x file.

Exporting "workbook.xls" to PDF will return in just one file: "workbook.pdf", but it might result in 3 files when exporting to HTML: "workbook.htm", "workbook_image1.png" and "workbook.css"

So we need a way to name the different generated files. By default FlexCelHtmlExport will take the name of the main HTML file as a parameter to the export, and generate the name of the images with the following pattern:

```
<ImagesFolder>\<htmlfilenamewithoutextension>_image<imagenumber>.<im
ageext>
```

This pattern might change, but all in all it gives a good default for images being created. You might use a GUID too as name for the generated images, by changing the **ImageNaming** property.

Now, you might want or need more control than this over the filenames created, and FlexCelHtmlExport gives you that with the **GetImageInformation** event, where you can specify exactly which filename you want for each image or even a stream for each one if you are saving for example the file to a database and not to a file system.

Remember that OnGetImageInformation will be called for every image in the xls file, even if they are not real images. For example, a chart will be rendered as an image, and so it will throw an OnGetImageInformation event. Also, if you have vertical text and the VerticalTextAsImages property is true, then this event will also be called when creating the images with the vertical text.

CSS files are other place where you might create extra files. The HTML standard allows embedding the CSS files inside the generated HTML or to use an external stylesheet, and so does FlexCelHtmlExport. If you decide to go for an external stylesheet, you need to tell FlexCelHtmlExport where to place it, by providing a **TCssInformation** class or a CSS filename to the export methods, that it can use to figure out where to save it.

Embeding the generated HTML inside your own pages

Other thing you that is different in generating HTML from other file formats, is that you might want to get only a part of the html file and not all of it. For example, imagine you want to embed an xls file inside your existing company page. You will not want the whole html file (since embedding <html> tags inside other <html> tags is not valid HTML), but only the part

between the <body> tags. And you will want to have the <head> part of the file in a separated place, so you can merge it with the <head> tags in your site.

With FlexCelHtmlExport, you can use the **PartialExportAdd** method and the **TPartialExportState** class for this.

You start by creating a new TPartialExportState instance, where FlexCel will store all the information it needs to create your files.

Then, you call FlexCelHtmlExport.PartialExportAdd() for each sheet or file you want to add to the HTML file, using the same TPartialExportState instance as parameter.

Once you have added all the sheets and files you want, you can use the methods in TPartialExportState to get the individual parts of the HTML file.

Customizing the generated HTML

General Customization

Customization is a very important part of creating HTML files, since differently from xls or pdf files, HTML files normally have to be integrated with an existing site. This means the style of the generated pages must match the general look and feel of the whole website, and we tried not to cut any corners in letting you do so.

Customizing the raw HTML

There is one property, ExtraInfo where you can add extra HTML in different parts of the document. For example, you can use this property to add META tags to your file including the keywords for searching.

Customizing the Sheet Selector

When exporting a workbook to different HTML files we offer three ways in which you can customize the tabs that represent each sheet:

1. You can get basic customization of colors and properties by changing the CssTags property in the TStandardSheetSelector class.

   The CssTags is a collection of macros that will be replaced in the style definition of the Selector, allowing to change the width of the selector (when placed at the right or at the top, the background color, etc). There is detailed information on the available properties you can change in the FlexCel reference.

2. If the basic customization is not enough, you can completely redefine the css properties in a TStandardSheetSelector class. There is a general property for the styles that apply to all the position in the selector (CssGeneral) and then customized properties for all the other positions.

   For example, you could set border black for all CssTabs with CssGeneral style, and then redefine the border as blue when the tab is on the left by using the CssWhenLeft property.

   You can use the built-in macros when defining your CSS properties and you can even define your own. For example, you could define:

   ```
   CssWhenLeft.Main = "float: left; width: <#width>";
   ```

   And then set

   ```
   TStandardSheetSelector.CssTags["width"] ="100px"
   ```

   As explained, you can even define your own macros. For example you could use

   ```
   CssWhenLeft.Main = "float: <#floatpos>;";
   ```

   And then add "floatpos" to the CssTags array. This way you will be later be able to change your own styles easily. The syntax for using a macro in a CSS definition is "<#macroname>", and then you need to add it to the CssTags array.

3. Normally methods 1) and 2) should be enough to handle most needs, but if you need a completely new way to display your tags, you can just define your own SheetSelector by deriving it from the class TSheetSelector. You will need to override the abstract methods in this class to provide the behavior you need. In fact, this is the way TStandardSheetSelector is defined, and you can look at its code when creating your own SheetSelector class.

Customizing the Fonts

Fonts used in the HTML file are the same that the ones used in the xls file, and this might bring some compatibility issues if you expect your file to be shown in a website where it can be opened by people all over the world.

So it is recommended that you stick to standard fonts, like Arial or Times New Roman on the xls files you want to export. But if the files already exist and have non standard fonts, you

can use the HtmlFont event to convert the fonts to standard typefaces. You can search for an example on this in the Export HTML demo.

Important note: The quote character FlexCel uses in style tags is the single quote ('). So, if you need to quote your font names because they have spaces, use a double quote (") so the style declaration is not affected.

For example, a style declaration in FlexCel might be: style = 'font-decoration: "my font"; '

You need to use double quotes around "my font" so it does not end the style declaration.

Export to HTML and FlexCel recalculation

Look at the notes in "Using FlexCelPdfExport" on the recalculating issues, this applies to HTML too.

Using Relative Hyperlinks

You can create links in Excel, both inside cells or on images, and they will be exported to html. But there is a small issue in that in Excel you can only enter absolute URLs. For example, you can enter:

"http://www.tmssoftware.com/docs/index.htm", but you cannot enter just docs/index.htm. If you want your websites to be "site independent", you need to replace those URLs with relative ones.

You can do this using the "BaseUrl" property. If for example you define"BaseUrl" to be "http://www.tmssoftware.com/", whenever a link starts with that text, the text will be removed. In our example, the link:

"http://www.tmssoftware.com/docs/index.htm" will be converted to

"docs/index.htm"