Part 2 - Creating a Cover sheet

Table of Contents

Introduction

This is Part 2 of the two-part tutorial series Extended Sales Summary scenario. It is recommended that you complete Part 1 - Creating a Dynamic Template before starting this section.

Following the Sample
There is a downloadable ExcelWriter_Basic_Tutorials.zip with completed templates and code. The completed example of the template is available under *templates/part2_template.xlsx*. The code for this part of the tutorial can be found in *Part2.aspx.cs*.

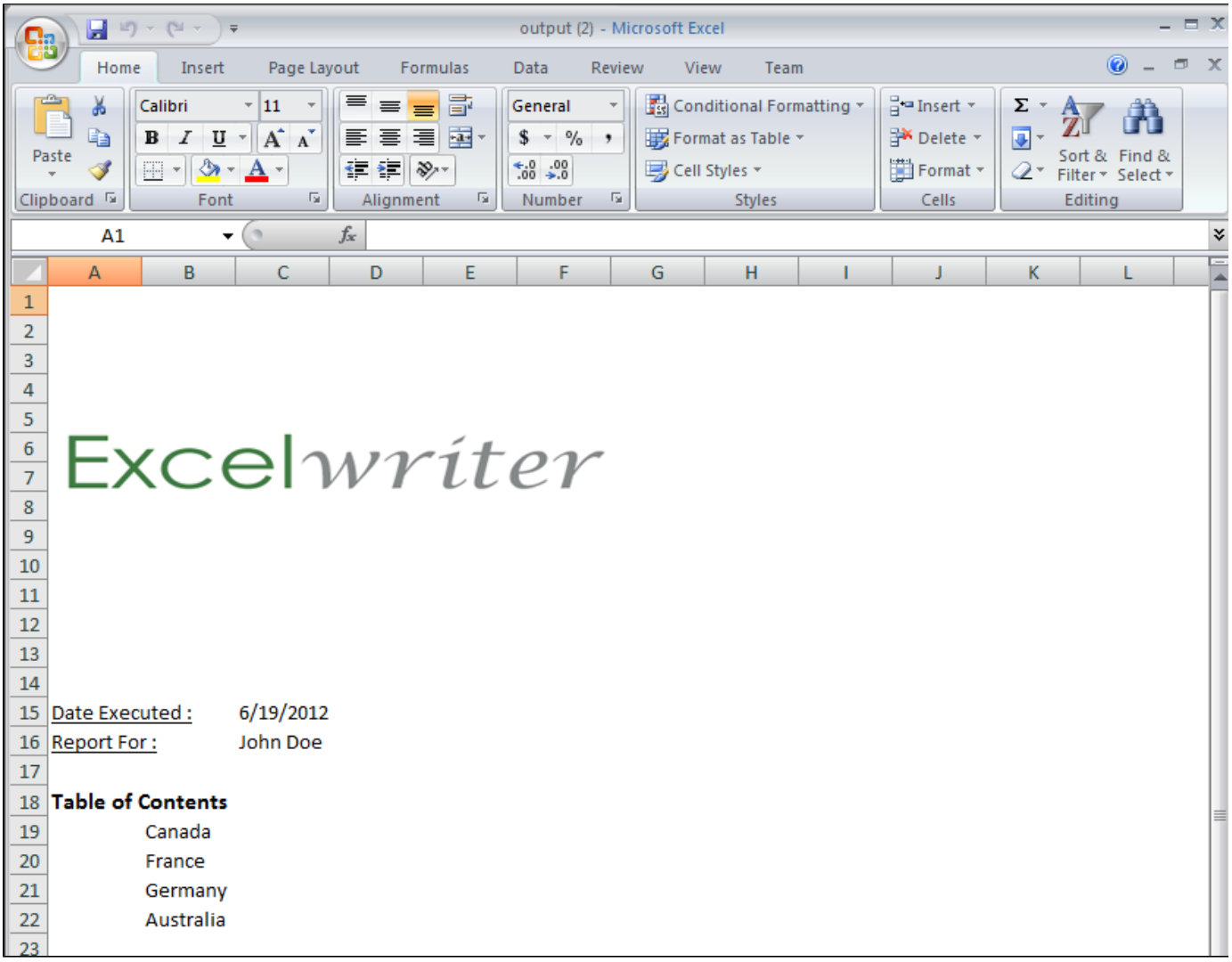
This part focuses on using `ExcelApplication` to create and a coversheet that an image, hyperlinks, and formatted text. It also includes binding data with `ExcelTemplate`.

Writing the Code

Writing the code for the cover sheet comes in two parts. the first part uses `ExcelApplication` to create a template coversheet; and the second part uses `ExcelTemplate` to bind the appropriate data to the data markers on the cover sheet.

Creating the coversheet with ExcelApplication

We want to create a cover sheet that looks like the following Excel worksheet:



Following the Sample Code
The code for this part of the tutorial can be found in *Part2.aspx.cs*

Setup

1. Define a method to contain the `ExcelApplication` code to create a new worksheet and customize it. In the sample, there is an `AddCoverSheet()` method that holds the code for the `ExcelApplicattion` code in this part of the tutorial.

```
protected void AddCoverSheet()
{

}
```

2. You should have already completed Part 1 of this tutorial. To include the `AddCoverSheet()` method, just add a call in `GenerateTemplate()`

```
protected void GenerateTemplate()
{

 xla = new ExcelApplication();

 wb = xla.Open(Page.MapPath(@"templates\template.xlsx"));

 for (int i = 0; i < selectedCountries.Count; i++)
 {
  wb.Worksheets.CopySheet(wb.Worksheets[0], wb.Worksheets.Count,
selectedCountries[i]);
 }

 wb.Worksheets["SimpleTemplate"].Visibility = Worksheet.SheetVisibility.Hidden;

 wb.Worksheets[1].Select();

 /*********Part 2*************/
 AddCoverSheet();
 /************************/

}
```

Adding the cover sheet

1. In `AddCoverSheet`, create a new worksheet called "Summary" at the beginning of the workbook with `Worksheets.CreateWorksheet`:

```
Worksheet ws = wb.Worksheets.CreateWorksheet("Summary", 0);
```

2. Select the "Summary" worksheet as the worksheet that will be active when the workbook opens with `Worksheet.Select()`.

```
ws.Select();
```

3. Hide the gridlines in the summary worksheet using `Worksheet.ShowGridlines`. By default, this property is set to `true`.

```
ws.ShowGridlines = false;
```

Inserting an image

1. The image will be inserted in cell A1. Adjust the height of row 1 to accommodate the image by retrieving the `RowProperties` object of row 1:

```
ws.GetRowProperties(0).Height = 90;
```

Note: The row height is set in units of 1/72 of an inch.

2. In the example, the user can select an image to insert into the file. The path of that image is dynamically retrieved and stored in variable `imagePath`.

```
string imagePath = RBImage.SelectedItem.Value;
```

3. Create an `Anchor` in cell A1 with `Worksheet.CreateAnchor()`. Set the offsets to 0.

```
Anchor anc = wb.Worksheets[0].CreateAnchor(0, 0, 0, 0);
```

4. Then insert the `Picture` object using `Pictures.CreatePicture`.

```
Picture logo = ws.Pictures.CreatePicture(Page.MapPath(@imagePath), anc);
```

Writing Text

1. Set the `Cell.Value` of cell A3 to the string "Date Executed :". In cell A4, set `Cell.Value` to "Report For:". In cell A6, set the `Cell.Value` to "Table of Contents".

```
ws.Cells["A3"].Value = "Date Executed :";
ws.Cells["A4"].Value = "Report For :";
ws.Cells["A6"].Value = "Table of Contents";
```

2. Insert the data markers for the "Date Executed" and "Report For" data, which will be imported using `ExcelTemplate`.

```
ws.Cells["C3"].Value = "%%=WebFormData.Date";
ws.Cells["C4"].Value = "%%=WebFormData.Name";
```

3. Next we will add hyperlinks to each of the worksheets in the workbook. To do this, we will use Excel's native `HYPERLINK` formula to point to the other worksheets. The format for the `HYPERLINK` formula will be:

`=HYPERLINK("#DestinationSheet!DestinationCell", "Display Text")`

This will be set to the `Cell.Formula` property.

Iterate through the country sheet names using a for loop, excluding the "Summary" and hidden "SimpleTemplate" sheets, which are the first two worksheets in the workbook.

```
for (int i = 2; i < wb.Worksheets.Count; i++)
{
    string sheetName = wb.Worksheets[i].Name.ToString();
    ws.Cells[16 + i, 1].Formula = "=HYPERLINK(\"#"+sheetName+"!A1\",
\""+sheetName+"\")";
}
```

Adding Styles to the Text

For more about using styles in ExcelWriter, see Effective Use of Styles.

1. Create a `GlobalStyle` using the `Workbook.CreateStyle()` method. This style will be applied to the labels on the summary sheet.

```
GlobalStyle labels = wb.CreateStyle();
```

2. Set the `Font.Bold` to `true` and `Font.Size` to 12.

```
labels.Font.Bold = true;
labels.Font.Size = 12;
```

3. Apply the labels style to the label cells with `Cell.ApplyStyle(Style)`.

```
ws.Cells["A3"].ApplyStyle(labels);
ws.Cells["A4"].ApplyStyle(labels);
ws.Cells["A6"].ApplyStyle(labels);
```

4. Set the `Font.UnderlineStyle` of cell A6 to be `UnderlineStyle.Single`.

```
ws.Cells["A6"].Style.Font.Underline = Font.UnderlineStyle.Single;
```

5. The final code for `AddCoverSheet()` should be:

```
protected void AddCoverSheet()
{
            Worksheet ws = wb.Worksheets.CreateWorksheet("Summary", 0);
            ws.Select();
            ws.ShowGridlines = false;

            /*******Inserting the Image********/


            ws.GetRowProperties(0).Height = 90;

            string imagePath = RBImage.SelectedItem.Value;

            Anchor anc = wb.Worksheets[0].CreateAnchor(0, 0, 0, 0);
            Picture logo = ws.Pictures.CreatePicture(Page.MapPath(@imagePath), anc);

            /*******Writing values*************/

            ws.Cells["A3"].Value = "Date Executed :";
            ws.Cells["A4"].Value = "Report For :";
            ws.Cells["A6"].Value = "Table of Contents";

            ws.Cells["C3"].Value = "%%=WebFormData.Date";
            ws.Cells["C4"].Value = "%%=WebFormData.Name";

            for (int i = 2; i < wb.Worksheets.Count; i++)
            {
                string sheetName = wb.Worksheets[i].Name.ToString();
                ws.Cells[7 + i, 1].Formula = "=HYPERLINK(\"#" + sheetName +
                    "!A1\", \"" + sheetName + "\")";
            }

            /******Adding Styles to Text*******/

            GlobalStyle labels = wb.CreateStyle();
            labels.Font.Bold = true;
            labels.Font.Size = 12;

            ws.Cells["A3"].ApplyStyle(labels);
            ws.Cells["A4"].ApplyStyle(labels);
            ws.Cells["A6"].ApplyStyle(labels);

            ws.Cells["A6"].Style.Font.Underline = Font.UnderlineStyle.Single;

}
```

Binding the coversheet data with ExcelTemplate

1. In Part 1, we defined `PopulateTemplate()`, which binds all of the data to the worksheet. We will add the calls to bind data to the summary sheet here.

2. In the sample, the user can provide their own recipient name. Retrieve this from the web form.

```
string recipient = TextBox1.Text;
```

3. Create an string array for the header values and a string array for the column names.

`ExcelTemplate` can be bound to numerous types of .NET data structures, some of which have built in column names, such as the DataTable. When working with arrays, which don't have built in column names, you have to define the column names in a separate string array.

```
string[] coverData = {recipient, DateTime.Now.Date.ToString("M/dd/yyyy")};
string[] coverMarkers = { "Name", "Date" };
```

3. Bind this row of data to the summary sheet with `ExcelTemplate.BindRowData`. `BindRowData()` binds a single row of data to the template, but the data markers in the template do not need to be in a single row.

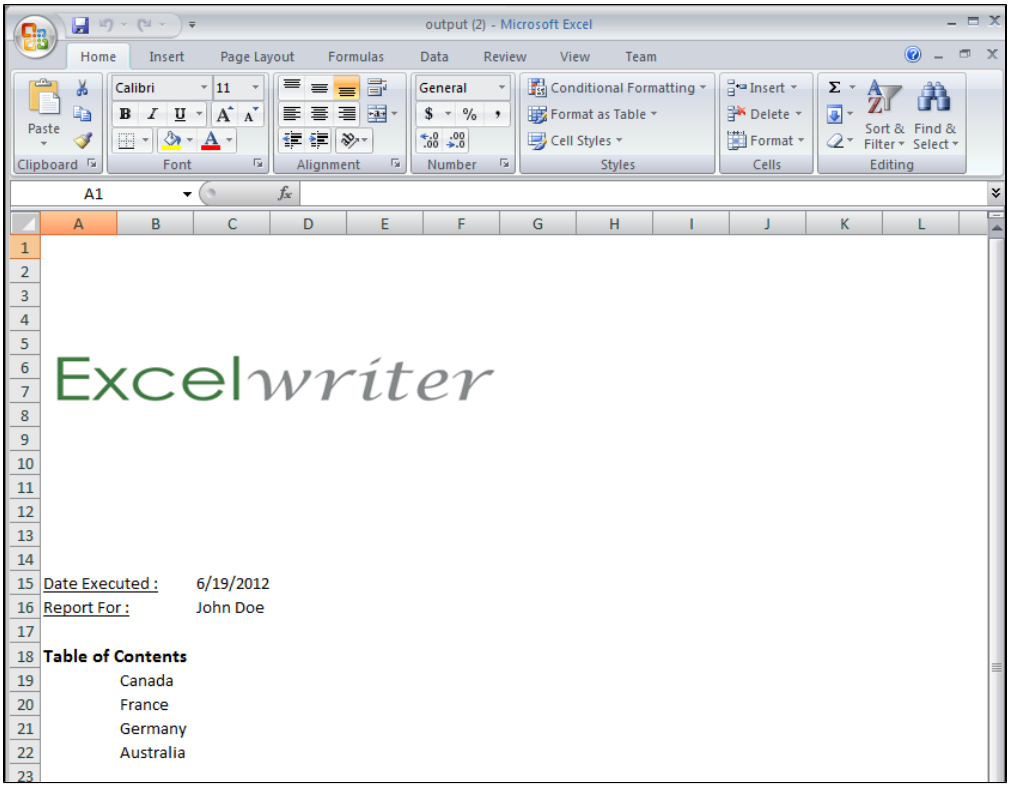
```
xlt.BindRowData(coverData, coverMarkers, "WebFormData",
xlt.CreateDataBindingProperties());
```

4. The final code for `PopulateTemplate()`, including the code from Part 1 should look like this:

```
protected void PopulateTemplate()
{
            xlt = new ExcelTemplate();
            xlt.Open(xla, wb);

            xlt.RemoveExtraDataMarkers = true;

            DataBindingProperties dataBindProps;
            for (int i = 0; i < selectedCountries.Count; i++)
            {
                string country = selectedCountries[i];

                dataBindProps = xlt.CreateDataBindingProperties();

                dataBindProps.WorksheetName = country;

                string[] headerValues = { "FY 2008", "Foreign Trade Division", country
};
                string[] headerNames = { "FiscalYear", "TradeDivision", "Country" };

                xlt.BindRowData(headerValues, headerNames, "Header", dataBindProps);

                DataTable dts = GetCSVData(Page.MapPath("//data//" + country +
"5.csv"));
                DataTable dts2 = GetCSVData(Page.MapPath("//data//" + country +
"All.csv"));

                xlt.BindData(dts, "Top", dataBindProps);
                xlt.BindData(dts2, "Details", dataBindProps);
            }

            /**********Part 2***********/
            string recipient = TextBox1.Text;

            string[] coverValues = { recipient,
DateTime.Now.Date.ToString("M/dd/yyyy") };
            string[] coverNames = { "Name", "Date" };

            xlt.BindRowData(coverValues, coverNames, "WebFormData",
                xlt.CreateDataBindingProperties());

            /*************************/

            xlt.Process();
            xlt.Save(Page.Response, "Output.xlsx", false);
}
```

5. Now run your code.

Here is an example of what the form will look like.



Notice that the countries in the Table of Contents are hyperlinked to the corresponding sheets in the workbook.

Downloads

You can download the code for the Extended Sales Summary here.

- ExcelWriter_Basic_Tutorials.zip