Part 1 - Getting Started

Table of Contents

Setting up the Template

About templates and data markers

An ExcelWriter **template** is an Excel file that contains ExcelWriter **data markers**. A **data marker** is a cell value beginning with %%= that specifies a database column, variable, or array to insert into the spreadsheet column. Data markers are added to a worksheet in Excel and then bound to data sources in code. ExcelWriter populates the data markers with values from the data sources when the code is executed.

ExcelWriter templates are typically created as standard Excel files (XLS, XLSX, XLSM), although "template" formats (XLT, XLTX, XLTM) are also supported by ExcelWriter.

Data marker syntax

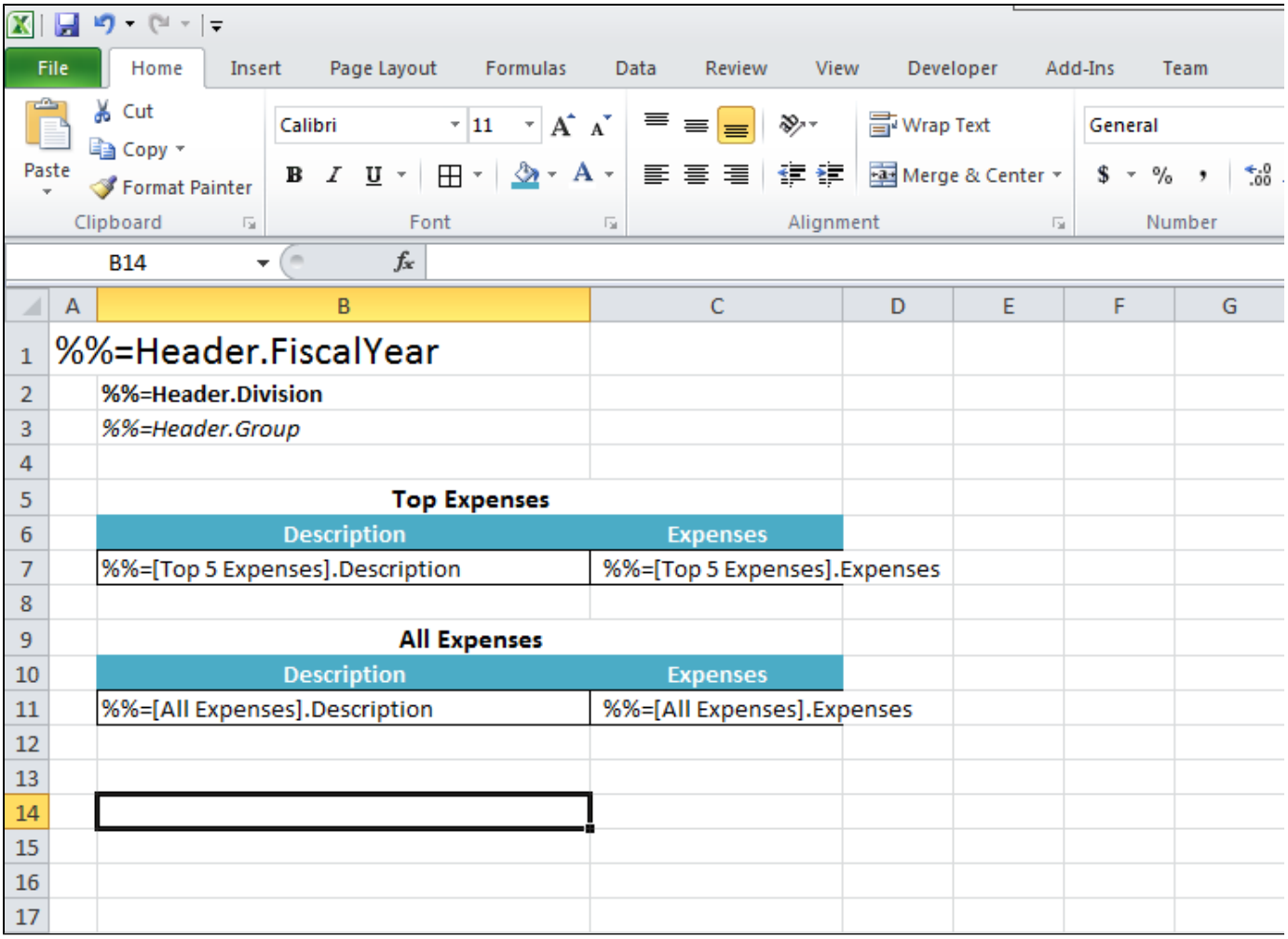
The basic syntax for a data marker is `%%=[DataSourceName].[ColumnName]`, where `DataSourceName` is the name of the data source and `ColumnName` is the name of the column in the data source. You need to follow these rules when naming data markers:

- Names must begin with a letter (A-Z, a-z)
- The brackets (`[ ]`) are optional, but must be used when the data marker contains spaces or Unicode characters
  - The following is a list of characters allowed in data marker names without brackets: ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890_
- Names must exactly match the names in the data source.
  - If a column in a database is 'Street Address', the `ColumnName` must be `[Street Address]` to account for the space. Similarly, if a data source name is "DataSource1", the data marker name must be `DataSource1` or `[DataSource1]`.

For more specific information about creating data markers, see Creating Data Markers

Adding data markers to the the template

The final template will look something like this:



Note: At the top of the worksheet, we will display the fiscal year, the company division and group. Below will be 2 tables: one to show the top 5 expenses and another to show all the expenses.

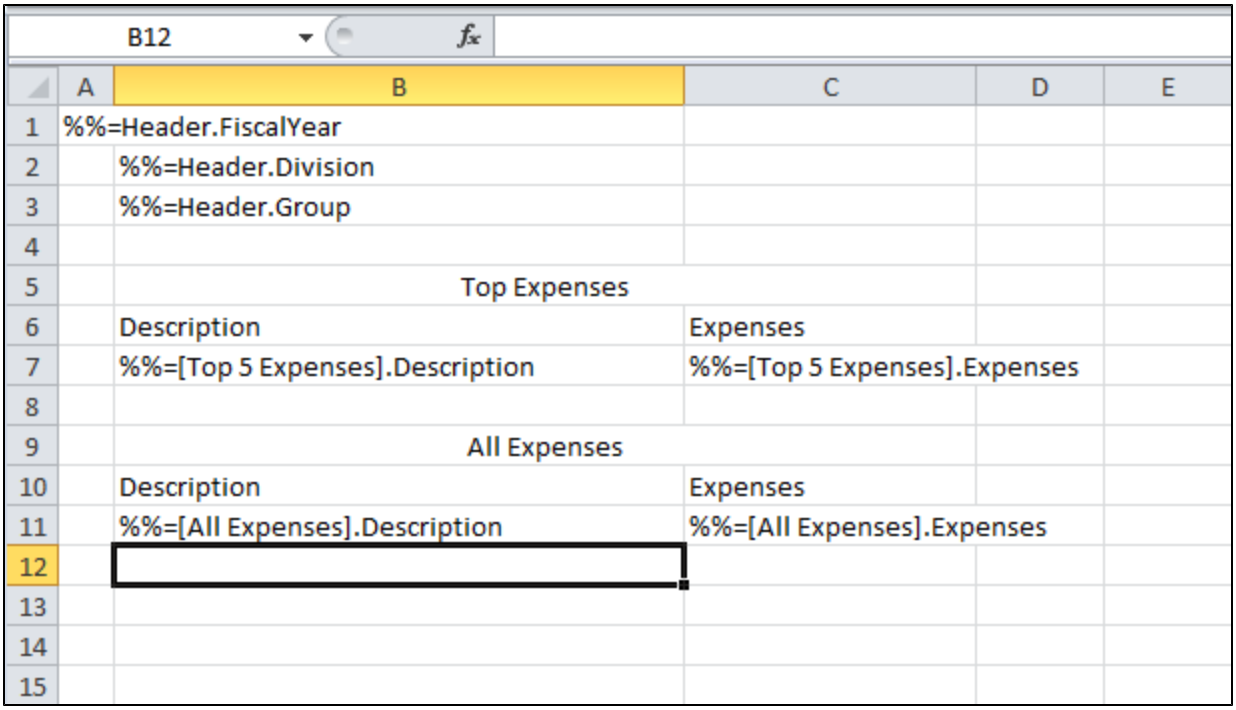
> **Following the Sample Code**
> In the downloadable ExcelWriter_Basic_Tutorials.zip under *SimpleExpenseSummary*, there is a completed template file located in *SimpleExpenseSummary/templates/part1_template.xlsx*.

1. Start with a blank .xlsx file. Save the file as template.xlsx.

2. Add some data markers for Replace Fiscal Year, Division, and Group. These values will be a single row of a data set called "Header". The column names will be "FiscalYear", "Division" and "Group".

3. Next add data markers for Top Expenses and All Expenses, which will be table header columns:

- The data source name for Top Expenses will be "Top 5 Expenses" with column names "Description" and "Expenses".
- The data source name for All Expenses will be "All Expenses" with the same column names.
- Since the data source names have spaces, the data markers need to be in brackets.



	A	B	C	D	E
1		%%=Header.FiscalYear			
2		%%=Header.Division			
3		%%=Header.Group			
4					
5		Top Expenses			
6		Description	Expenses		
7		%%=[Top 5 Expenses].Description	%%=[Top 5 Expenses].Expenses		
8					
9		All Expenses			
10		Description	Expenses		
11		%%=[All Expenses].Description	%%=[All Expenses].Expenses		
12					
13					
14					
15					

We're done adding the data markers, so next we'll hook the template up to some data before we do any formatting.

Adding an ExcelWriter Reference in Visual Studio

> **Following the Sample Code**
> In the sample code, the reference to *SoftArtisans.OfficeWriter.ExcelWriter.dll* has already been added to the *SimpleExpenseSummary* p roject.

Create a .NET project and add a reference to the ExcelWriter library.

1. Open Visual Studio and create a .NET project.
    - The sample code uses a web application.
2. Add a reference to SoftArtisans.OfficeWriter.ExcelWriter.dll
    - SoftArtisans.OfficeWriter.ExcelWriter.dll is located under **Program Files > SoftArtisans > OfficeWriter > dotnet > bin**

Writing the Code

> **Following the Sample Code**
> There is a sample web application page `Part1.aspx` and code behind `Part1.aspx.cs` available in the **SimpleExpenseSummary/** d irectory that shows the completed code.

1. Include the SoftArtisans.OfficeWriter.ExcelWriter namespace in the code behind

```
using SoftArtisans.OfficeWriter.ExcelWriter;
```

2. In the method that will actually run the report, instantiate the `ExcelTemplate` object.

```
ExcelTemplate XLT = new ExcelTemplate();
```

3. Open the template file with the `ExcelTemplate.Open` method.

```
XLT.Open(Page.MapPath("//templates//part1_template.xlsx"));
```

4. Create a `DataBindingProperties` object. Although we won't be changing any of the binding properties, a `DataBindingProperties` is a required parameter in all `ExcelTemplate` data binding methods.

```
DataBindingProperties dataProps = XLT.CreateDataBindingProperties();
```

5. Create an `object` array for the header values and a `string` array for the column names.

`ExcelTemplate` can be bound to numerous types of .NET data structures: single variables, arrays (1-D, jagged, multi-dimensional), `DataSet`, `DataTable`, `IDataReader` etc. The source of the data can come from anywhere.

Some of the aforementioned structures 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.

```
//This report is for FiscalYear: FY 2004, Division: Canadian Division, Group: Research
and Development
object[] valuesArray = { "FY 2004", "Canadian Division", "Research and Development" };

//The column names are FiscalYear, Division, Group
string[] columnNamesArray = { "FiscalYear", "Division", "Group" };
```

6. Use the `ExcelTemplate.BindRowData` method to bind the header data to the data markers in the template file (`%%=Header.FiscalYear`, `%%=Header.Division`, `%%=Header.Group`).

`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(valuesArray, columnNamesArray, "Header", dataProps);
```

7. Get the data for the Top 5 Expenses and All Expenses data sets.

Following the Sample
In the sample project, we are parsing CSV files with query results, rather than querying a live database. The CSV files are available under the *data* directory. There is a copy of the CSV parser, `GenericParsing.dll` in the *bin* directory of the project `GetCSVData` is defined in *Part1.aspx.cs* in a `region` marked *Utility Methods*.

These calls are to a helper method `GetCSVData` that parses the CSV files and returns a `DataTable` with the values.

```
DataTable dtTop5 = GetCSVData(Page.MapPath("//data//Part1_Top5Expenses.csv"));
DataTable dtAll = GetCSVData(Page.MapPath("//data//Part1_AllExpenses.csv"));
```

If you are following in your own project and would like to parse the CSV files as well, you will need to:

- Add a reference to `GenericParsing.dll`
- Include `GeneringParsing` at the top of your code.
- Add the `GetCSVData` method that can be found in the sample code.

8. Use `ExcelTemplate.BindData` to bind the data for the Top 5 Expenses and All Expenses data sets.

Recall that the data source names (`[Top 5 Expenses]`, `[All Expenses]`) need to match the data marker names exactly.

```
XLT.BindData(dtTop5, "Top 5 Expenses", dataProps);
XLT.BindData(dtAll, "All Expenses", dataProps);
```

9. Call `ExcelTemplate.Process()` to import the data into the file.

```
XLT.Process();
```

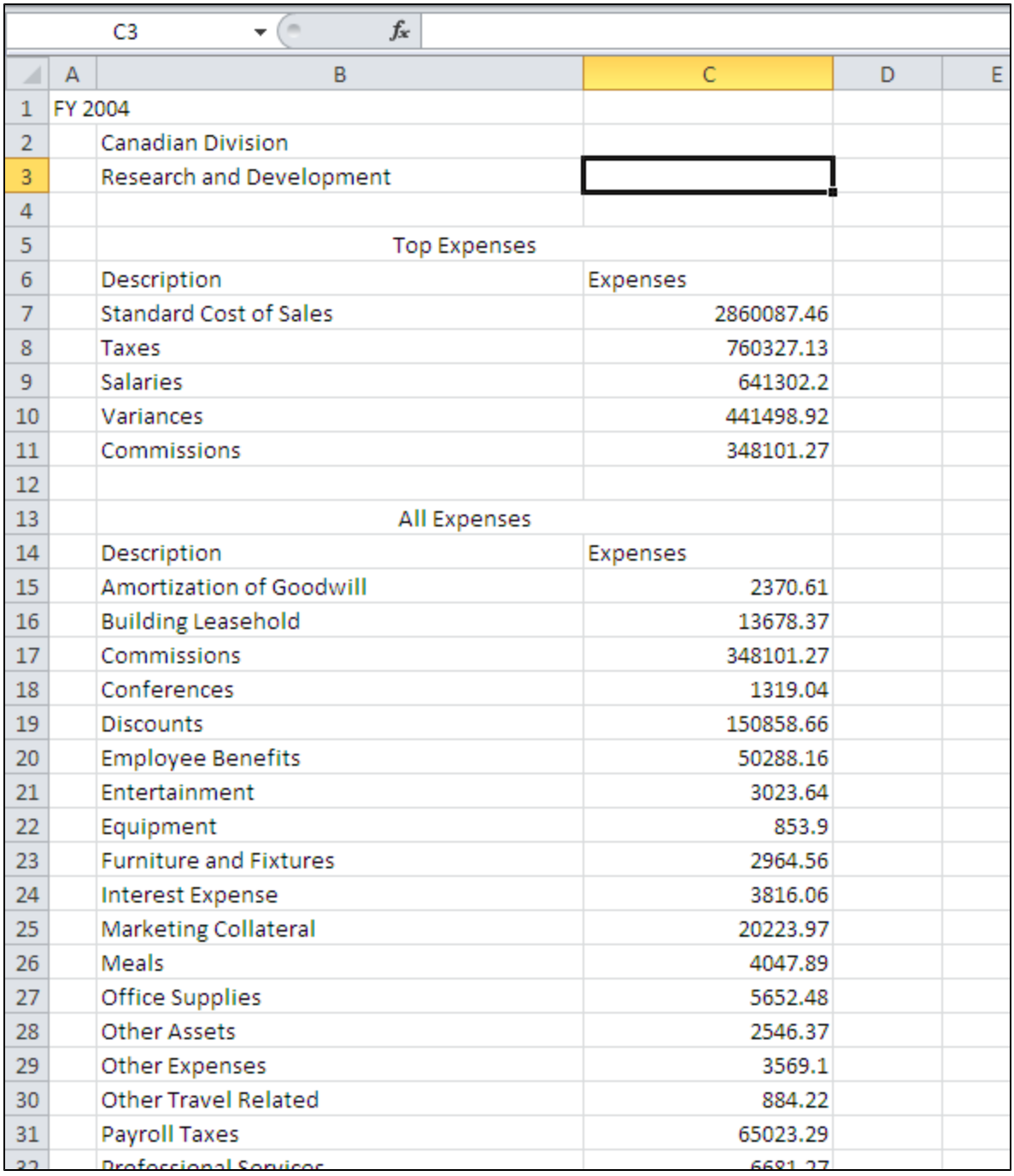
10. Call `ExcelTemplate.Save` to save the output file.

`ExcelTemplate` has several output options: save to disk, save to a stream, stream the output file in a page's `Response` inline or as an attachment.

```
XLT.Save(Page.Response, "Part1_Output.xlsx", false);
```

11. Run your code.

Here is an example of the output from the sample code:

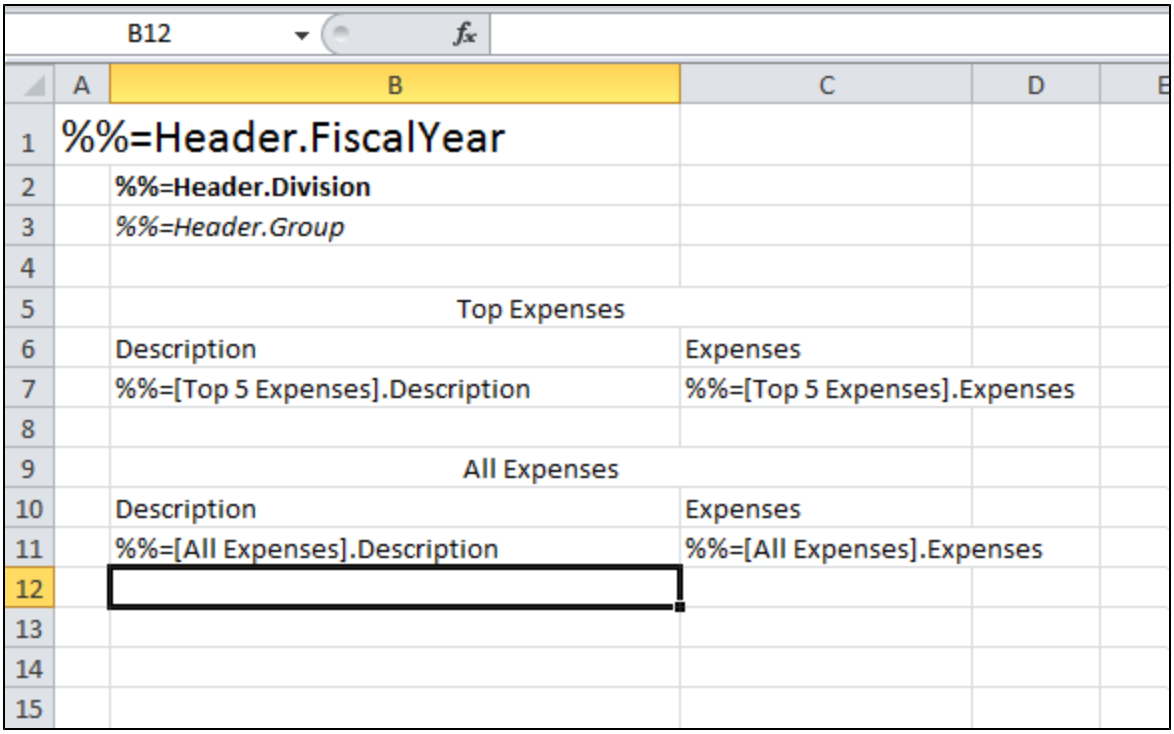


	A	B	C	D	E
1	FY 2004				
2		Canadian Division			
3		Research and Development			
4					
5		Top Expenses			
6		Description	Expenses		
7		Standard Cost of Sales	2860087.46		
8		Taxes	760327.13		
9		Salaries	641302.2		
10		Variances	441498.92		
11		Commissions	348101.27		
12					
13		All Expenses			
14		Description	Expenses		
15		Amortization of Goodwill	2370.61		
16		Building Leasehold	13678.37		
17		Commissions	348101.27		
18		Conferences	1319.04		
19		Discounts	150858.66		
20		Employee Benefits	50288.16		
21		Entertainment	3023.64		
22		Equipment	853.9		
23		Furniture and Fixtures	2964.56		
24		Interest Expense	3816.06		
25		Marketing Collateral	20223.97		
26		Meals	4047.89		
27		Office Supplies	5652.48		
28		Other Assets	2546.37		
29		Other Expenses	3569.1		
30		Other Travel Related	884.22		
31		Payroll Taxes	65023.29		
32		Professional Services	6681.27		

Formatting cells with data markers

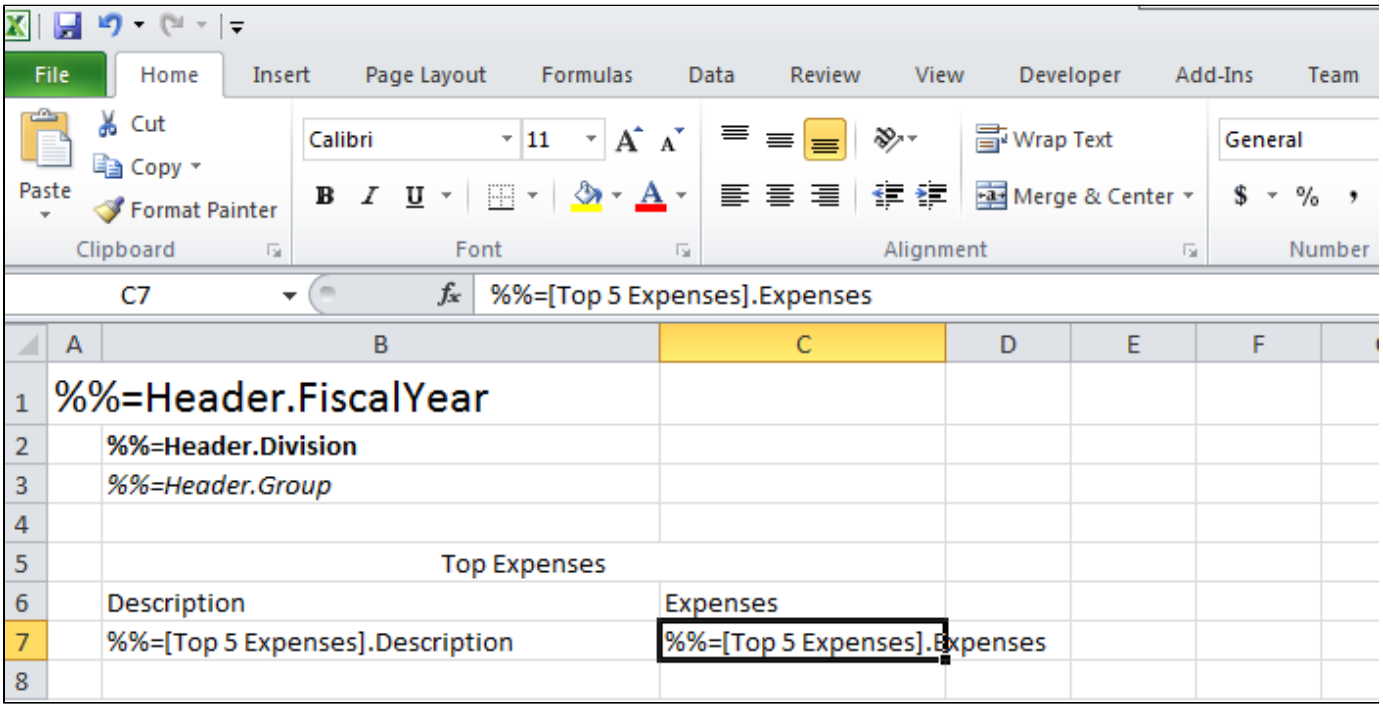
> Data markers take the formatting and style properties of the cell that they are in. This means if a data marker is bold, then the value that replaces the data marker will be bold as well.

1. In the screen shot below we have made the `%%=Header.FiscalYear` cell font size 18, `%%=Header.Division` is **bold**, and `%%=Header.Group` is *italic*.

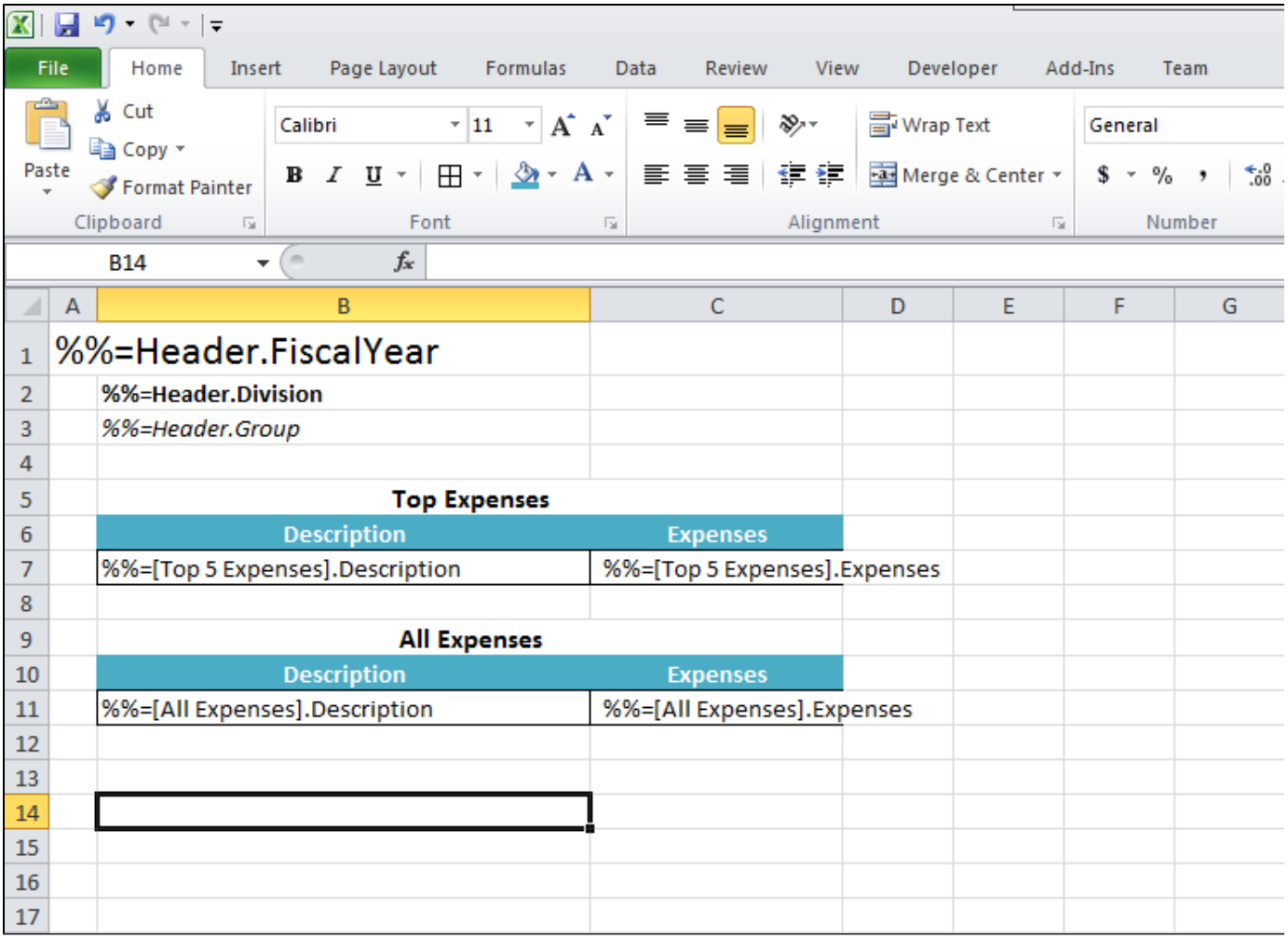


When importing multiple rows of data, ExcelWriter will insert a new row in the worksheet for each row of data, starting from the row with the data markers. Each of the new rows will take on the styles and formatting of the cells that contain the data markers. For more details on this behavior see How ExcelWriter Inserts Rows.

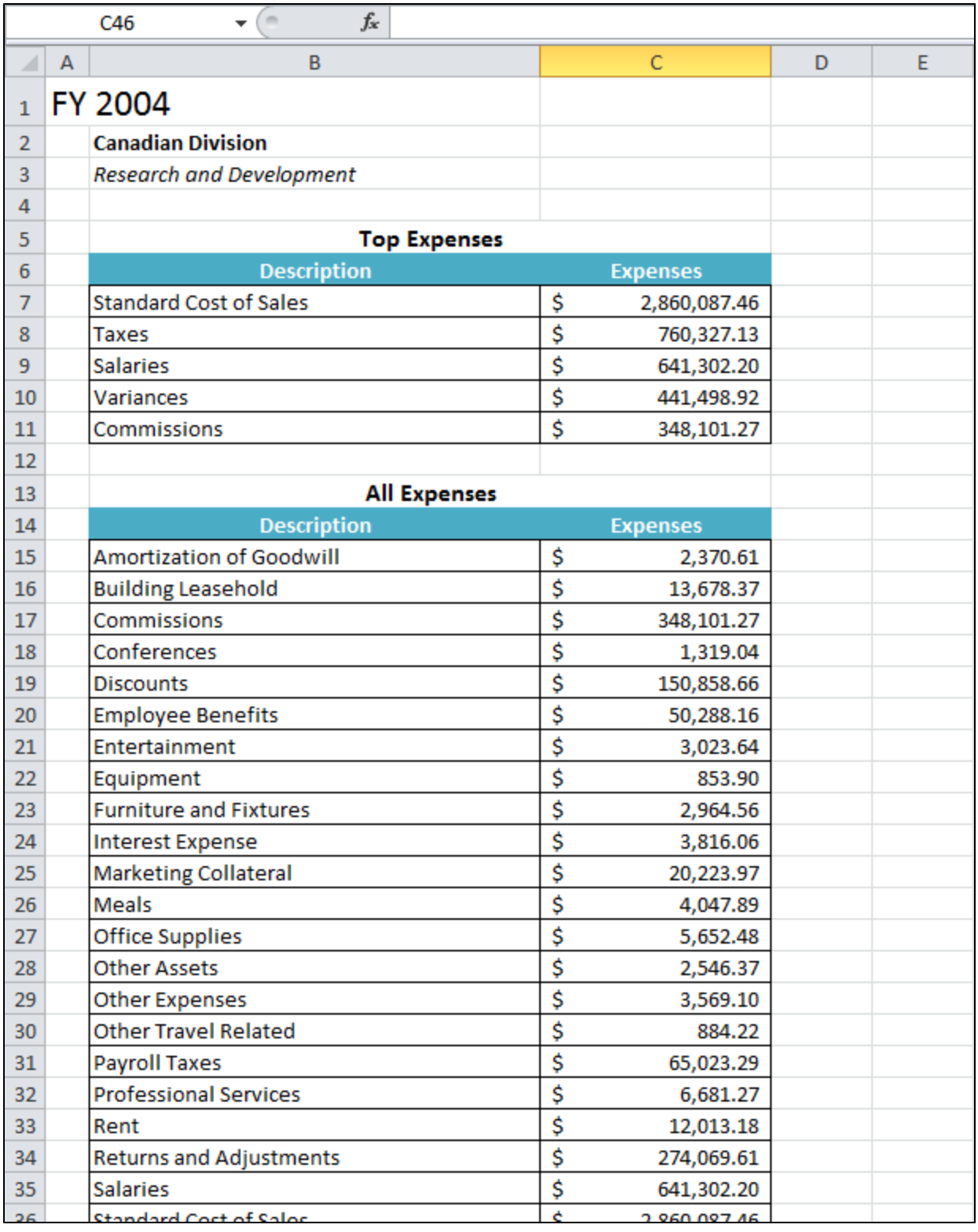
2. Since the 'Expenses' data will be currency values, add a currency number formatting to the cells containing the `Expenses` data markers. This number formatting will be repeated for each row of data that is inserted.



3. Add some borders to the cells in the Top Expenses and All Expenses tables. Then format the column headers as desired. Below is a screen shot of the final template:



4. Run the code with the updated template file. Here's a screenshot of the output with all the formatting applied:



	A	B	C	D	E
1	FY 2004				
2		**Canadian Division**			
3		*Research and Development*			
4					
5		**Top Expenses**			
6		Description	Expenses		
7		Standard Cost of Sales	$ 2,860,087.46		
8		Taxes	$ 760,327.13		
9		Salaries	$ 641,302.20		
10		Variances	$ 441,498.92		
11		Commissions	$ 348,101.27		
12					
13		**All Expenses**			
14		Description	Expenses		
15		Amortization of Goodwill	$ 2,370.61		
16		Building Leasehold	$ 13,678.37		
17		Commissions	$ 348,101.27		
18		Conferences	$ 1,319.04		
19		Discounts	$ 150,858.66		
20		Employee Benefits	$ 50,288.16		
21		Entertainment	$ 3,023.64		
22		Equipment	$ 853.90		
23		Furniture and Fixtures	$ 2,964.56		
24		Interest Expense	$ 3,816.06		
25		Marketing Collateral	$ 20,223.97		
26		Meals	$ 4,047.89		
27		Office Supplies	$ 5,652.48		
28		Other Assets	$ 2,546.37		
29		Other Expenses	$ 3,569.10		
30		Other Travel Related	$ 884.22		
31		Payroll Taxes	$ 65,023.29		
32		Professional Services	$ 6,681.27		
33		Rent	$ 12,013.18		
34		Returns and Adjustments	$ 274,069.61		
35		Salaries	$ 641,302.20		
36		Standard Cost of Sales	$ 2,860,087.46		

Note: The formatting has been applied to the values that replaced the data markers, including the data sets with multiple rows. Also note that the Top 5 Expenses and All Expenses tables have expanded to accommodate the new rows of data (i.e. All Expenses was pushed down when the Top 5 Expenses data was imported).

Final Code

```
using SoftArtisans.OfficeWriter.ExcelWriter;
...
ExcelTemplate XLT = new ExcelTemplate();

XLT.Open(Page.MapPath("//templates//part1_template.xlsx"));

DataBindingProperties dataProps = XLT.CreateDataBindingProperties();

object[] valuesArray = { "FY 2004", "Canadian Division", "Research and Development" };
string[] columnNamesArray = { "FiscalYear", "Division", "Group" };

XLT.BindRowData(valuesArray, columnNamesArray, "Header", dataProps);

DataTable dtTop5 = GetCSVData(Page.MapPath("//data//Part1_Top5Expenses.csv"));
DataTable dtAll = GetCSVData(Page.MapPath("//data//Part1_AllExpenses.csv"));

XLT.BindData(dtTop5, "Top 5 Expenses", dataProps);
XLT.BindData(dtAll, "All Expenses", dataProps);

XLT.Process();

XLT.Save(Page.Response, "Part1_Output.xlsx", false);
```

Downloads

You can download the code for the Basic ExcelWriter Tutorials as a Visual Studio solution, which includes the Simple Expense Summary.

- ExcelWriter_Basic_Tutorials.zip

Next Steps

Continue on to Part 2: Working with Formulas