

Part 3 - Combine Reports with CopySheet

Table of Contents

- [Introduction](#)
 - [Getting Started](#)
- [Adding an ExcelWriter Reference in Visual Studio](#)
- [Writing the Code](#)
 - [Merging template together with CopySheet](#)
 - [Binding Data to the Merged Template](#)
- [Final Code](#)
- [Downloads](#)

Introduction



This is Part 3 of the three-part tutorial series [Financial Report](#) scenario. It is recommended that you complete [Part 1 - Using Modifiers and Ordinal Syntax](#) and [Part 2 - Using Styles and Formatting](#) before starting this section.

This tutorial also assumes a basic understanding of how to open and manipulate Excel files with `ExcelApplication`.

Getting Started

This section covers using `ExcelApplication` to join two Worksheets into one Workbook. These two worksheets were created in [Part 1](#) and [Part 2](#) of this tutorial.



Following the Sample Code

In the downloadable [ExcelWriter_Basic_Tutorials.zip](#), there are completed template files located in `CompleteFinancialReport/templates`.

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 `CompleteFinancialReport` project.

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

Merging template together with CopySheet

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

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

2. Globally declare the `ExcelTemplate`, `ExcelApplication`, and `Workbook` objects.

```
ExcelApplication XLA;  
ExcelTemplate XLT;  
Workbook MERGEDWB;
```

3. Define a method that will open the two template files and merge them together with [CopySheet](#). In this case, the method is called `MergeTemplates()`.

```
protected void MergeTemplates()  
{  
  
}
```

4. In `MergeTemplates()`, instantiate the [ExcelApplication](#) object.

```
XLA = new ExcelApplication();
```

5. Open the workbook that the worksheets will be copied into. For this example, the second template will be copied into the first template workbook.

```
MERGEDWB = XLA.Open(Page.MapPath("~/templates//Part1_Financial_Template.xlsx"));
```

6. Open the second workbook that will be merged into the first.

```
Workbook wbOther =  
XLA.Open(Page.MapPath("~/templates//Part2_Financial_Template.xlsx"));
```

7. Copy the worksheet from the second template into the first workbook. [Worksheet.Name](#) can be used to pass along the original name of the worksheet ("Percent by Quarter"). The worksheet should be copied to the end of the current worksheet collection (index 1).

```
MERGEDWB.Worksheets.CopySheet(wbOther[0], 1, wbOther[0].Name);
```

Binding Data to the Merged Template

1. Create a method to handle binding and processing the data import for the combined template. For this example, the method is called `BindDataToMergedTemplate()`.

```
protected void BindDataToMergedTemplate()  
{  
  
}
```

2. Instantiate [ExcelTemplate](#) and open the template file with [ExcelTemplate.Open\(ExcelApplication, Workbook\)](#) to open directly from the pre-existing `ExcelApplication` instance.

```
XLT = new ExcelTemplate();  
XLT.Open(XLA, MERGEDWB);
```

3. Use the code from [Part 1](#) and [Part 2](#) to bind data to the template. Note that both templates used the same data set, so the calls to bind the data only need to be made once.



If there are duplicate data markers in a workbook, ExcelWriter will import the matching data to all the data markers, across all the worksheets in the workbook.

```
//Create data binding properties for ExcelTemplate
DataBindingProperties bindingProps = XLT.CreateDataBindingProperties();

//Create the array of header values. This example only binds a single item
string[] headerValues = { "2011" };

//Create the array of header names.
string[] headerNames = { "FiscalYear" };

//Get the datatables to be bound into the template.
//This example uses CSV data, but the data can come from
//anywhere
DataTable dtAssets = GetCSVData("//data//Assets.csv");
DataTable dtLosses = GetCSVData("//data//Losses.csv");
DataTable dtOther = GetCSVData("//data//Other.csv");

//Bind the header row data
XLT.BindRowData(headerValues, headerNames, "Header", bindingProps);

//Bind the datatables
XLT.BindData(dtAssets, "Assets", bindingProps);
XLT.BindData(dtLosses, "Losses", bindingProps);
XLT.BindData(dtOther, "Other", bindingProps);

//Process to import the data to the file
XLT.Process();

XLT.Save(Page.Response, "Part3_Output.xlsx", false);
```

Final Code

```
using SoftArtisans.OfficeWriter.ExcelWriter;
using GenericParsing; //See Part 1 for more information about the CSV parser
...
//Declare globals
ExcelApplication XLA;
ExcelTemplate XLT;
Workbook MERGEDWB;

//Merges the reports and binds data to the templates
protected void RunReport()
{
    //Join the two reports using CopySheet
    MergeTemplates();

    //bind the data for the newly modified file
    BindDataToMergedTemplate();
}
```

```

}

protected void JoinReports()
{
    //Instantiate ExcelApplication
    XLA = new ExcelApplication();

    //Open the first template
    MERGEDWB = XLA.Open(Page.MapPath("~/templates//Part1_Financial_Template.xlsx"));

    //Open the second template
    Workbook wbOther =
XLA.Open(Page.MapPath("~/templates//Part2_Financial_Template.xlsx"));

    //Call CopySheet
    MERGEDWB.Worksheets.CopySheet(wbOther[0], 1, wbOther[0].Name);
}

protected void BindTemplateData()
{
    XLT = new ExcelTemplate();

    //Open the template file
    XLT.Open(XLA, MERGEDWB);

    //Create data binding properties for ExcelTemplate
    DataBindingProperties bindingProps = XLT.CreateDataBindingProperties();

    //Get the datatables to be bound into the template.
    //This example uses CSV data, but the data can come from
    //anywhere
    DataTable dtAssets = GetCSVData("~/data//Assets.csv");
    DataTable dtLosses = GetCSVData("~/data//Losses.csv");
    DataTable dtOther = GetCSVData("~/data//Other.csv");

    //Bind the datatables
    XLT.BindData(dtAssets, "Assets", bindingProps);
    XLT.BindData(dtLosses, "Losses", bindingProps);
    XLT.BindData(dtOther, "Other", bindingProps);

    /*This section only applies to the first template. It will be ignored by the
    second template, since there are no markers to bind to.*/

    //Create the array of header values. This example only binds a single item
    string[] headerValues = { "2011" };

    //Create the array of header names.
    string[] headerNames = { "FiscalYear" };

    //Bind the header row data
    XLT.BindRowData(headerValues, headerNames, "Header", bindingProps);

    //end

    //Process to import the data to the file
    XLT.Process();
}

```

```
XLT.Save(Page.Response, "Part3_Output.xlsx", false);
```

```
}
```

Downloads

You can download the code for the Financial Report here.

- [ExcelWriter_Basic_Tutorials.zip](#)