Hello World using ExcelTemplate

Table of Contents

Hello World with ExcelTemplate

ExcelWriter's ExcelTemplate approach allows you to write data to a template file that contains Data Markers. The data markers tell ExcelWriter where to bind specific sets of data. This tutorial will show you the basics on how to dynamically insert data into a worksheet using ExcelTemplate by taking custom text from a web form textbox and inserting it into a template file.

Setting up the Template File

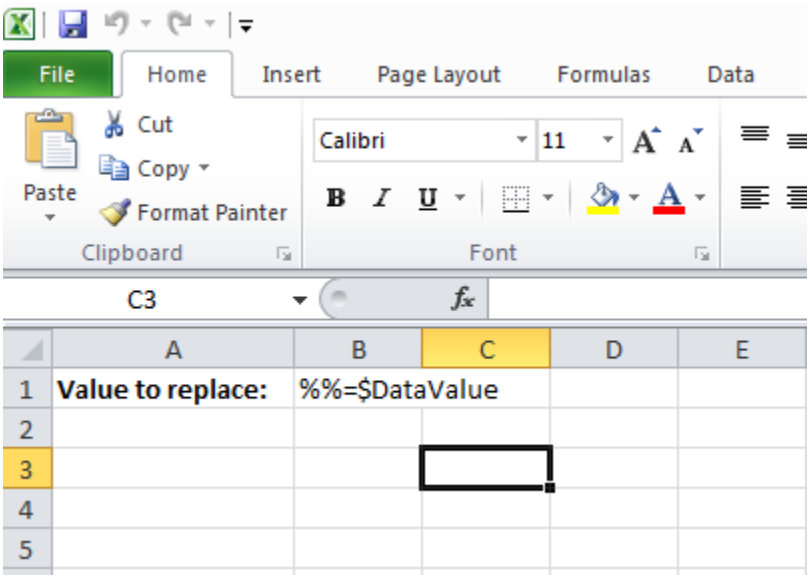
1. Create a new .XLSX file. Save it as template.xlsx.

   > In the *Hello World* sample web application, the completed template file is located in \templates\Hello World.xlsx.

2. We are going to bind a single string value to a cell in a template file. To do this, we will first need to add a data marker to the cell where we want the value to appear.

   All ExcelWriter data markers are prefaced with '**%%=**' and the additional '**$**' sign means that the data source for this data marker is 1-dimensional (e.g. 1-dimensional array or single value). 'DataValue' is the data marker ID we'll use to bind the data to this data marker.

   Below is a screenshot of the completed template file from the *Hello World* sample:

3. Now the template file is done. Next is writing the code to bind the string value to the data marker.

Writing the code

> In the sample code, the completed file is: ExcelTemplate_HelloWorld.aspx.[cs/vb]. The corresponding web form is

ExcelTemplate_HelloWorld.aspx.

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

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

```
 Imports SoftArtisans.OfficeWriter.ExcelWriter
```

2. Instantiate the ExcelTemplate object.

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

```
Dim XLT As New ExcelTemplate()
```

3. Open the template file with ExcelTemplate.Open.

   The ExcelTemplate object corresponds to a single template file, so a given ExcelTemplate instance can only have one template file open.

```
XLT.Open(Page.MapPath("templates\\Hello World.xlsx"));
```

```
XLT.Open(Page.MapPath("templates\Hello World.xlsx"))
```

4. Create a DataBindingProperties object

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

```
Dim DataProps As DataBindingProperties = XLT.CreateDataBindingProperties()
```

   The DataBindingProperties object can be used to change the behavior of how data is imported. For example, if we were importing multiple rows of data, we can use the DataBindingProperties.MaxRows property to limit the number of rows that are imported. In this sample, we won't be changing any of the import properties, but we still need the DataBindingProperties object to bind data.

5. Get the data and call ExcelTemplate.BindCellData() to bind the data to the data marker

```
string value = DataValueBox.Text.Trim();
XLT.BindCellData(value, "DataValue", DataProps);
```

```
Dim value As String = DataValueBox.Text.Trim()
XLT.BindCellData(value, "DataValue", DataProps)
```

In this sample, we're pulling the single value from the text box on the web form. Since we're binding a single value, we use BindCellData() and specify the data marker ID. Note that we need to pass the DataBindingProperties object, even though none of the DataBindingProperties are active.

6. Call ExcelTemplate.Process() to insert the data into the file.

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

```
XLT.Process()
```

ExcelTemplate.Process() handles everything relating to inserting the data into the file. If we were importing multiple rows of data, Process() would handle inserting the new physical rows into the Excel worksheet.

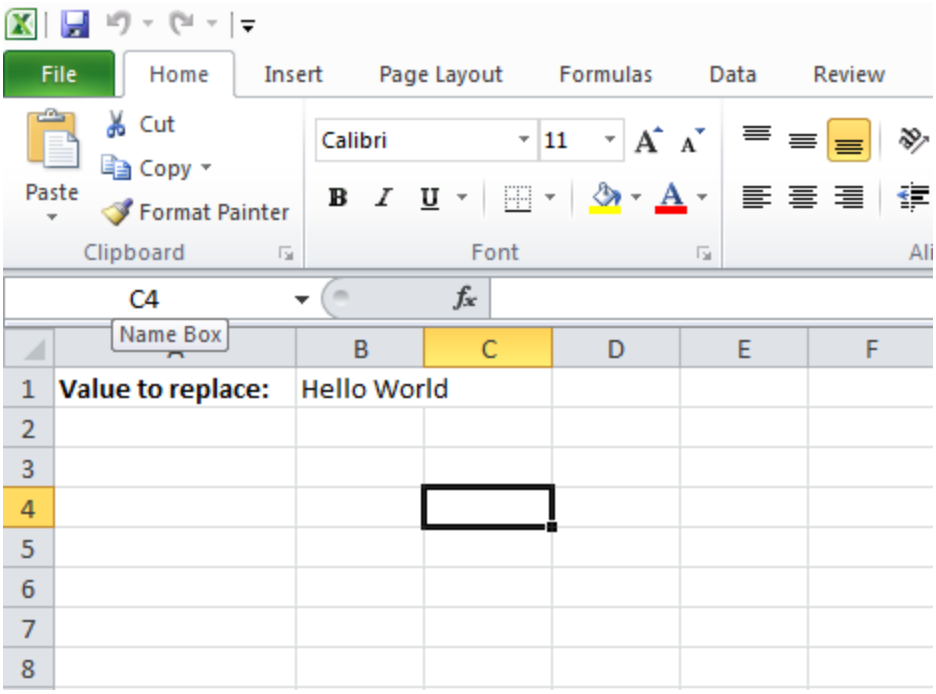
7. Save the output

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

```
XLT.Save(Page.Response, "Output.xlsx", False)
```

There are several options for ExcelTemplate.Save including: save to disk, save to memory stream, stream back to the client inline, and stream back to the client as an attachment. In this case, we're streaming the workbook back to the client as an attachment. Also, ExcelWriter does not convert between file formats, so it is important that the file extension on the output file matches the file extension of the original template file.

8. Go to the web form page, **ExcelTemplate_HelloWorld.aspx**, to try out the sample. In the output file, you will see that the data marker has been replaced with the custom text entered in the form.



Congratulations, you have completed Hello World for ExcelTemplate!

Final Code

```
using SoftArtisans.OfficeWriter.ExcelWriter;
...
ExcelTemplate XLT = new ExcelTemplate();
XLT.Open(Page.MapPath("Hello World.xlsx"));
DataBindingProperties DataProps = XLT.CreateDataBindingProperties();
string value = DataValueBox.Text.Trim();
XLT.BindCellData(value, "DataValue", DataProps);
XLT.Process();
XLT.Save(Page.Response, "Output.xlsx", false);
```

```
Imports SoftArtisans.OfficeWriter.ExcelWriter
...
Dim XLT As New ExcelTemplate()
XLT.Open(Page.MapPath("templates\Hello World.xlsx"))
Dim DataProps As DataBindingProperties = XLT.CreateDataBindingProperties()
Dim value As String = DataValueBox.Text.Trim()
XLT.BindCellData(value, "DataValue", DataProps)
XLT.Process()
XLT.Save(Page.Response, "Output.xlsx", False)
```

Downloads

You can download the code for the hello world tutorial as a Visual Studio solution.

- [ExcelWriter_HelloWorldC#.zip]
- [ExcelWriter_HelloWorldVB.zip]