Part 1 - Mail Merges and NEXT fields

Table of Contents

Intro

⚠ This tutorial assumes a basic understanding of the `WordTemplate` object and behavior when importing multiple rows of data. If you are not familiar with these concepts, please go through the Sales Invoice tutorial first.

⚠ There is only 1 part to this tutorial.

The `WordTemplate` object is used for template-driven document generation. This object opens a WordWriter template file, populates it with data from a specified data source, and generates a new Word document. The `WordTemplate` object provides the ability to import multiple rows of data by repeating sections of a document for each row, or using the entire document as a repeatable section.

This tutorial covers how to use `WordTemplate.SetMailMerge` to use the entire document as a repeatable section and then how to use NEXT fields to populate a sheet of business labels, over multiple pages.

Business Label Template

ⓘ **Following the Sample Code**
In the downloadable WordWriter_Basic_Tutorials.zip there is a completed template file located in *BusinessLabels/templates/Business_Label_Template.docx*.

The template file looks like this:

There is a table with fields for an employee's FirstName, MI, LastName, Phone, Email, Street, City, State, and ZipCode. These fields are repeated for an entire page in the document. The goal is to populate all of the business labels, but spill onto multiple pages as necessary.

Unlike with repeat blocks that are populated with `WordTemplate.SetRepeatBlock`, which require bookmarks to determine what part of the document is repeated, `WordTemplate.SetMailMerge` treats the entire document as a repeat block, so it is not necessary to add bookmarks when using `SetMailMerge`.

Adding a WordWriter Reference in Visual Studio

> ⓘ **Following the Sample Code**
> In the sample code, the reference to `SoftArtisans.OfficeWriter.WordWriter.dll` has already been added to the *BusinessLabels* web application project.

To create a .NET project and add a reference to the WordWriter library:

1. Open Visual Studio and create a .NET project.
2. Add a reference to the SoftArtisans.OfficeWriter.WordWriter.dll
   - `SoftArtisans.OfficeWriter.WordWriter.dll` is located under **Program Files > SoftArtisans > OfficeWriter > dotnet > bin**.

Writing the Code

> ⓘ **Following the Sample Code**
> The code for this tutorial can be found under *BusinessLabels/Part1.aspx.cs*.

This part of the tutorial will cover how to set up the WordTemplate code to bind the data to the template. This assumes a basic knowledge of WordTemplate and how to bind data to Word templates.

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

```
using SoftArtisans.OfficeWriter.WordWriter;
```

2. Create a new WordTemplate object and open the template file.

```
WordTemplate WT = new WordTemplate();
WT.Open(Page.MapPath("//templates//Business_Label_template.docx"));
```

3. Get the employee data for the business labels.

> ⓘ **Following the Sample Code**
> 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. The helper method GetCSVData is defined in *Part1.aspx.cs* in a region marked *Utility Methods.*

This call is to a helper method GetCSVData that parses the CSV files and returns a DataTable with the values.

```
DataTable dtBusinessLabels = GetCSVData("//data//BusinessLabelTutorialData.csv");
```

4. Use WordTemplate.SetMailMerge to bind the employee data to the template.

Recall that SetMailMerge treats the entire document as a repeat block. This means that the merge fields in the document will need to match the column names of the data source. If there are any stray merge fields in the document, WordWriter will throw an error.

```
WT.SetMailMerge(dtBusinessLabels);
```

5. Process and save the template file.

```
WT.Process();
WT.Save(Page.Response, "Business_Label_Output.docx", false);
```

The completed code should look like this:

```
//Instantiate a new WordTemplate object
WordTemplate WT = new WordTemplate();

//Open the template file
WT.Open(Page.MapPath("//templates//Business_Label_template.docx"));

//Get the order info datatable using GenericParser
DataTable dtBusinessLabels = GetCSVData("//data//BusinessLabelTutorialData.csv");

//Set the data sources to import a single row of data for each source
WT.SetMailMerge(dtBusinessLabels);

//Process to import the data to the template
WT.Process();

WT.Save(Page.Response, "Business_Labels_Output.docx", false);
```
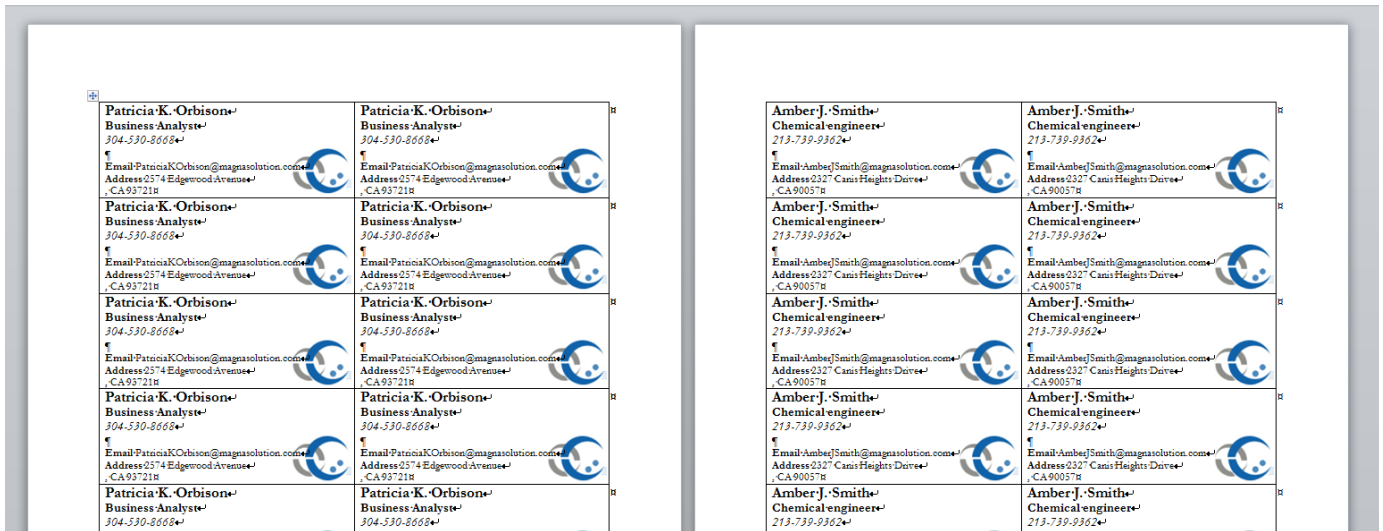
Run the code. You will notice that each employee has a page of business labels with that particular employee's data.

This is the intended `SetMailMerge` behavior, as the entire document was repeated for each row of data. In the next section we will incorporate next fields so that only one business label is made for each employee.

Adding NEXT fields

The NEXT fields change how WordWriter imports multiple rows of data. In regular repeat blocks, an entire portion of the document will be repeated for every row of data, or in the case of `SetMailMerge`, the entire document will be repeated. With NEXT fields, WordWriter will first look to see if it can populate a merge field that is preceded by a NEXT field before repeating the section.

This will allow us to populate through an entire page of business labels before moving on to a second page.

NEXT fields in the template

1. Place the cursor in the topmost right table cell. This is where the NEXT field will go.

«FirstName» «MI». «LastName»↵	«FirstName» «MI». «LastName»↵
«Position»↵	«Position»↵
«Phone»↵	«Phone»↵
¶	¶
Email· «Email»↵	Email· «Email»↵
Address· «Street»↵	Address· «Street»↵
«Town», «State»· «ZipCode»¤	«Town», «State»· «ZipCode»¤
«FirstName» «MI». «LastName»↵	«FirstName» «MI». «LastName»↵
«Position»↵	«Position»↵
«Phone»↵	«Phone»↵
¶	¶
Email· «Email»↵	Email· «Email»↵
Address· «Street»↵	Address· «Street»↵
«Town», «State»· «ZipCode»¤	«Town», «State»· «ZipCode»¤

2. Go to Insert > Quick Parts and select Field from the drop-down. Select *Next* from the list of available fields. Click OK to add the NEXT field.

3. Repeat this process for each business label on the page, except for the first business label. When you are done, press ALT + F9 to show all the NEXT fields to confirm.

| {·MERGEFIELD·FirstName·*· MERGEFORMAT}·{· MERGEFIELD·MI·*· MERGEFORMAT}.·{· MERGEFIELD·LastName·*· MERGEFORMAT}↵
{·MERGEFIELD··Position·*· MERGEFORMAT}↵
{·MERGEFIELD··Phone·*·MERGEFORMAT· }↵
¶
Email·{·MERGEFIELD··Email··*·MERGEFORMAT·}↵
Address·{·MERGEFIELD··Street··*·MERGEFORMAT· }↵
{·MERGEFIELD··Town··*·MERGEFORMAT·},·{· MERGEFIELD··State··*·MERGEFORMAT·}·{· MERGEFIELD··ZipCode··*·MERGEFORMAT·}¤ | {·NEXT·*·MERGEFORMAT}{· MERGEFIELD·FirstName·*· MERGEFORMAT}·{· MERGEFIELD·MI·*· MERGEFORMAT}.·{· MERGEFIELD·LastName·*· MERGEFORMAT}↵
{·MERGEFIELD··Position·*· MERGEFORMAT}↵
{·MERGEFIELD··Phone·*·MERGEFORMAT· }↵
¶
Email·{·MERGEFIELD··Email··*·MERGEFORMAT·}↵
Address·{·MERGEFIELD··Street··*·MERGEFORMAT· }↵
{·MERGEFIELD··Town··*·MERGEFORMAT·},·{· MERGEFIELD··State··*·MERGEFORMAT·}·{· MERGEFIELD··ZipCode··*·MERGEFORMAT·}¤ |
| {·NEXT·*·MERGEFORMAT}{· MERGEFIELD·FirstName·*· MERGEFORMAT}·{· MERGEFIELD·MI·*· MERGEFORMAT}.·{· MERGEFIELD·LastName·*· MERGEFORMAT}↵
{·MERGEFIELD··Position·*· MERGEFORMAT}↵ | {·NEXT·*·MERGEFORMAT}{· MERGEFIELD·FirstName·*· MERGEFORMAT}·{· MERGEFIELD·MI·*· MERGEFORMAT}.·{· MERGEFIELD·LastName·*· MERGEFORMAT}↵
{·MERGEFIELD··Position·*· MERGEFORMAT}↵ |

4. Press ALT + F9 to hide the field codes again. The template is now ready for NEXT fields.

Changes to the code for NEXT fields

By default, NEXT fields will be ignored by WordWriter. To enable NEXT fields, set the `WordTemplate.EnableNEXTFields` property. This property must be set before `WordTemplate.Open` is called.

```
WT.EnableNEXTFields = true;
```

Final Code

```
//Instantiate a new WordTemplate object
WordTemplate WT = new WordTemplate();

//NEXT fields will be ignored by default
//This must be set before Open() is called
WT.EnableNEXTFields = true;

//Open the template file
WT.Open(Page.MapPath("//templates//Business_Label_template.docx"));

//Get the order info datatable using GenericParser
DataTable dtBusinessLabels = GetCSVData("//data//BusinessLabelTutorialData.csv");

//Set the data sources to import a single row of data for each source
WT.SetMailMerge(dtBusinessLabels);

//Process to import the data to the template
WT.Process();

WT.Save(Page.Response, "Business_Labels_Output.docx", false);
```

Run the code. The business labels are now populated throughout the page, only moving to the second page after the first page is filled.

Downloads

You can download the code for the Basic WordWriter Tutorials as a Visual Studio solution, which includes the Business Labels tutorial.

- WordWriter_Basic_Tutorials.zip