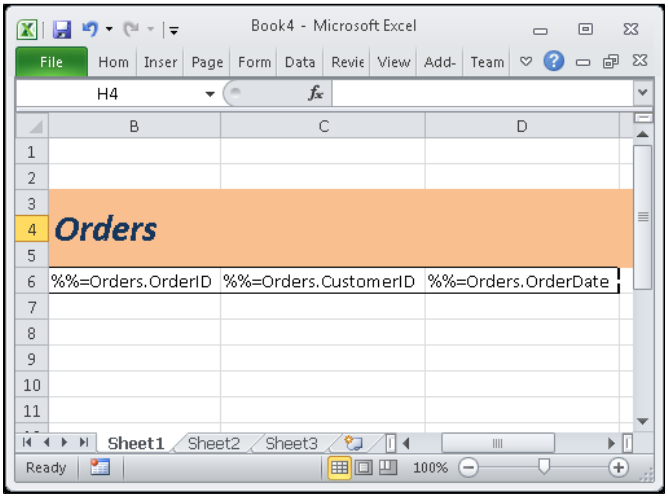
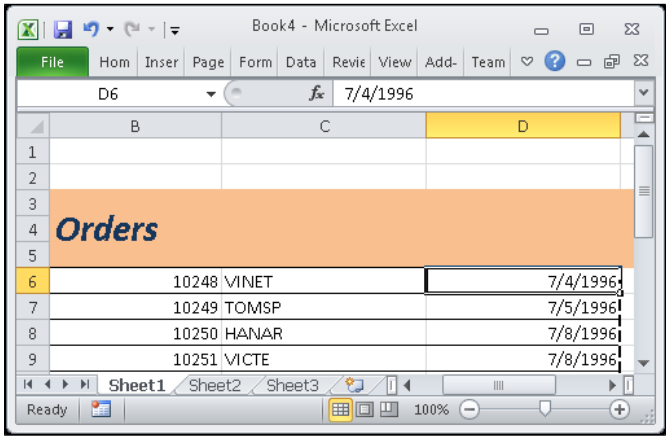
How to Use Templates

An ExcelWriter **template** is an Excel spreadsheet that contains ExcelWriter data markers. A data marker is a cell value beginning with **%%=** or **%%=$** that specifies a database column, variable, or array to insert in the spreadsheet column. A data marker may include modifiers.

Refer to the ExcelTemplate Code Samples to see live demonstrations of various ExcelTemplate applications. Click the **View Template** link on each sample page to see the template.

You can create an ExcelWriter template in Microsoft Excel or in script using the ExcelApplication object. Include data markers where you want to insert values. For example, if cell **B6** contains the data marker **%%=Orders.OrderID**, where Orders represents the Orders table in a database, ExcelWriter will import the **OrderID** column to column **B** in the spreadsheet.

Template	Spreadsheet generated from the template
	

A data marker binds in script to a data source which may be an array, DataTable, DataSet, or DataReader, and may include modifiers. Data source and field numbers are 1-based. If ExcelWriter encounters **%%=#0[.field]** or **%%=[DataSource.]#0**, an error will occur.

> ⚠ In ExcelWriter versions before 3.1, data source and field numbers were 0-based. If you upgraded from an earlier version, you may need to modify your code.

Data Marker Formats	Code
Database data marker	**%%=[DataSourceNameOrNumber.]FieldNameOrNumber[(modifier)]**
Variable data marker	**%%=$DataSourceNameOrNumber[(modifier)|(modifier)]**
One-Dimensional Array data marker	**%%=$DataSourceNameOrNumber[(modifier)|(modifier)]**
Two-dimensional Array data marker	**%%=$DataSourceNameOrNumber.ColumnNameOrNumber[(modifier)|(modifier)]**

Example

The following example generates a new spreadsheet from a template that contains two data markers: **%%=$RecipientName** and **%%=$RecipientCompany**. ExcelWriter sets the data sources for the template data markers to two simple string variables.

The ExcelTemplate object represents the template Excel file.

ExcelTemplate is in the SoftArtisans.OfficeWriter.ExcelWriter namespace. The object can be referenced as SoftArtisans.OfficeWriter.ExcelWriter.ExcelTemplate. To minimize typing and errors, use an Import directive to import the namespace to the .aspx page and reference the object as ExcelTemplate without the namespace prefix. If you are coding directly in the .aspx page include the following after the Page directive:

```
<%@ Import Namespace="SoftArtisans.OfficeWriter.ExcelWriter" %>
```

If you are coding in the code-behind page, include a using or Imports statement at the top of the code-behind page:

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

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

To generate a new spreadsheet with ExcelWriter:

1. Create an instance of ExcelTemplate, for example:

```
ExcelTemplate xlt = new ExcelTemplate();
```

```
Dim xlt As New ExcelTemplate()
```

2. Call ExcelTemplate.Open() to open a template Excel file, for example:

```
xlt.Open(Application["templatepath"] +
@"\DataBinding\StringBindingTemplate.xls");
```

```
xlt.Open(Application("templatepath") & _"\DataBinding\StringBindingTemplate.xls")
```

   The Open method takes the file path and name of the template .xls file to open.

3. Use the ExcelTemplate.BindCellData() method to assign data sources to bind to the template's data markers, for example:

```
//--- Bind the variables to the template data markers
//--- %%=$RecipientName and %%=$RecipientCompany
xlt.BindCellData("J. Smith", "RecipientName", xlt.CreateDataBindingProperties());
xlt.BindCellData("SoftArtisans", "RecipientCompany",
xlt.CreateDataBindingProperties());
```

```
'--- Bind the variables to the template data markers
'--- %%=$RecipientName and %%=$RecipientCompany
xlt.BindCellData("J. Smith", "RecipientName", xlt.CreateDataBindingProperties())
xlt.BindCellData("SoftArtisans", "RecipientCompany",
xlt.CreateDataBindingProperties())
```

4. Call ExcelTemplate.Process() to populate the template's data markers with data source values:

```
xlt.Process();
```

```
xlt.Process()
```

The Process method enters data source values in the template's merge fields, and creates the output file (the new spreadsheet) in memory.

5. Call ExcelTemplate.Save to generate a new spreadsheet:

```
xlt.Save(Page.Response, "StringBinding.xls", false);
```

```
xlt.Save(Page.Response, "StringBinding.xls", false)
```

If you pass Save a Page.Response object, ExcelWriter will stream the generated file to the client. Save's second parameter specifies a name for the generated Excel file; this name will be displayed in the download dialog when the file is streamed to the browser. If the third parameter is set to true and the user chooses to open the file, the file will open in the browser window; if it is set to false (as in the example) and the user chooses to open the file, the file will open in Microsoft Excel.

ExcelWriter allows you to save the generated file on the server or stream it to the client. For more information, see ExcelTemplate Output Options.