1 Class Generator Users Guide

ClassGenerator is an easy to use, source code generator, which uses pre-formatted templates. The source code templates include tag-based commands, which are interpreted respectively by the parser.

The general procedure would be:

i. Select one or more templates.
ii. Select the tables that reside in one or more databases.
iii. Issue the command to generate the real source files.

The generated files could be VB (ver. 6.0 or earlier) classes or forms, VB .NET classes / forms, ASP pages, ASP .NET WebForms, etc. Actually the generated files could be of any type – the generator does not care / discriminate file types – as long as there is a template to parse.

Note: The parser does not execute syntax analysis on the template, so any miswritten templates lead to failed code generation.

2 Main Application Window

The main window holds all the information needed by *ClassGenerator* to execute and generate source files. Top to bottom there are the below GUI controls:

GUI Control	Description	Comments
Output File	The full path of the generated file. For multiple classes / templates this is the output directory. When generating multiple classes you can change each class' path by selecting it in the Classes List View and editing this textbox.	
Output File Browse Button [...]	Press the button to locate the target directory.	
Template File	The full path of the selected template file. If Use Multiple Templates is selected you can add more templates by simply pressing the Template File Browse Button [...] and selecting the needed template. The above procedure is followed once for every template. The selected templates are shown in the list below.	
Template File Browse Button [...]	Press the button to open the template selection dialog box.	
Use Multiple Templates	Check if using more than one template file. Default value unchecked.	

Create Separate Folders	Check if you want to create folders for each class in the Classes List View. Default value unchecked.	The folder will have the class' name. All generated files will have the name of the corresponding template file.
Remove Template File Button [-]	Removes the selected in the list template file. Applies only when Use Multiple Templates is checked.	
Class Name	The name of the selected class in the Classes List View.	Tag: CLASS_NAME
Table Name	The respective table name of the selected class in the Classes List View.	Tag: TABLE_NAME
Table Name Format Button [#]	Defines the pattern used to extract the class name from the table name. Default value tbl<%TABLE_NAME%>	e.g. if the table name is tblTest, the class name would be Test.
Project Name	The project name the class(es) belong. Usually needed by .NET code.	Tag: PROJECT_NAME
Options Button		
Classes List View	The classes exported by the Database Explorer. You can change the class name by editing the item entry name in the List View. Selecting an item fills the textboxes above (class name, table name, output file). Editing these textboxes refreshes class properties accordingly.	
Use Multiple Classes	Check if using more than one class. Default value checked.	
Properties List View	The properties of the selected class in the Classes List View. For each property you can see the database type, the (VB) mapped type and the length when appropriate.	
Add Property Button	Deprecated.	
Remove Property Button	Deprecated.	
Edit Property Button	Deprecated.	
Database Schema Button	Opens the Database Explorer to choose the tables to export as classes. Each export from Database Explorer clears the previously selected classes.	
Generate Classes Button	Runs the parser for each class in Classes List View,	

	for the selected template(s).	

The main window of ClassGenerator is shown below (**Figure 1 Class Generator Main Window**).

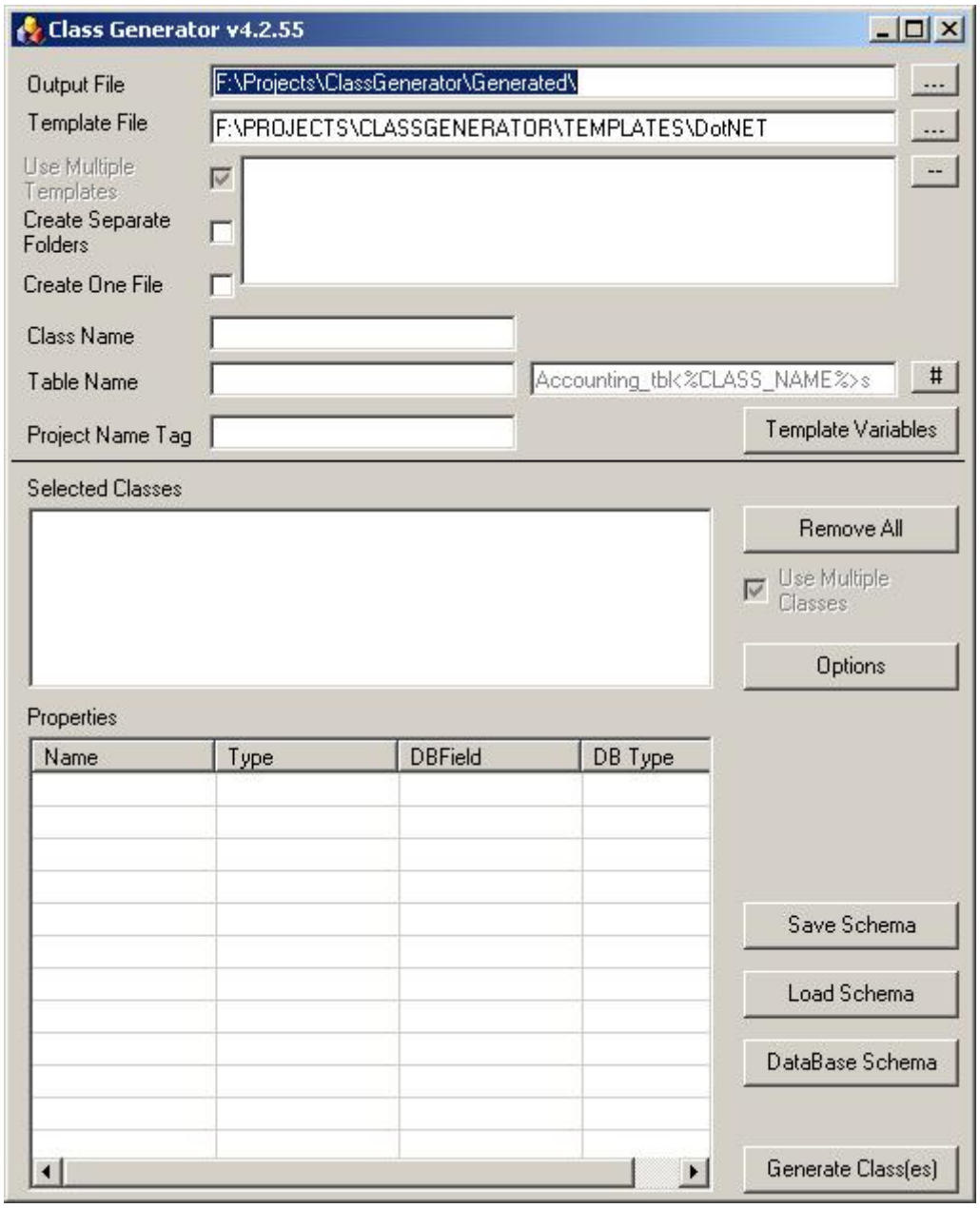

Figure 1 Class Generator Main Window

3 Generated File(s) Target Directory

You can define the directory where the generated files will be saved. You can either type the full path, or select the target directory using the browse dialog box. The browse dialog box appears by pressing the button besides the text box [**...**]. The select directory dialog box is shown below (**Figure 2 Select Directory Dialog Box**)

Figure 2 Select Directory Dialog Box

4 Template File(s) Browser

You can define the full path of the template to be used by the parser, or select it by using the browsing dialog box. To open the dialog box, press the button besides the textbox [**...**]. The system's open file dialog box appears (**Figure 3 Template Files Browse Dialog Box**).

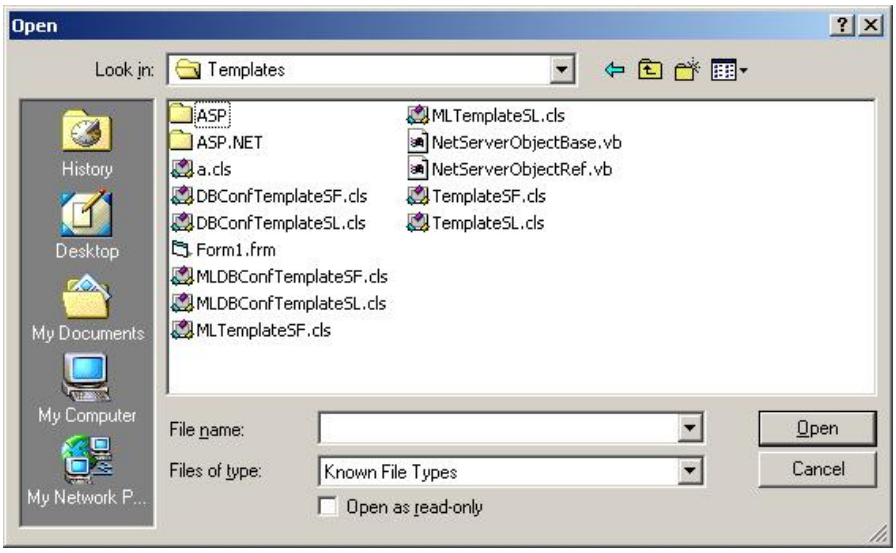

Figure 3 Template Files Browse Dialog Box

Only one file at a time can be selected. By default the dialog is set to filter files to known file types. Known file types are:

VB Class Files (*.cls)
VB Form Files (*.frm;*.frx)
Active Server Pages (*.asp)
VB .NET Class Files (*.vb)
VB .NET Form Files (*.vb;*.resx)

ASP .NET (*.aspx;*.aspx.resx;*.aspx.vb)
C# (*.cs)

However, one can select the no filter option All Files (*.*), to see all files in the selected directory.

4.1 Using Multiple Templates

If **Use Multiple Templates** is checked you can add multiple template files. Press the browse template files button [**...**], select one and press the OK button. The full path of the selected template file is added in the list box below the Template File text box. You can as many template files as you want. To remove an unwanted template file from the list, select the file and press the remove button, which is next to the list box [**-**].

If **Create Separate Folders** is checked (default value is unchecked) the generated files will be created under the defined output directory, in a folder named after the selected class. This means that for every item in the **Classes** list view a new folder will be created (if not already existing – existing directories will not loose any files). For every selected template file a file will be created with the name of that template file.

On the other hand, the generated file will take the name of the corresponding class and the extension of the template file. Since this is the case, when selecting more than one template files with the same extension, only one file will be generated, the one of the last template in the list. Actually all files will be created but the later will overwrite the previous.

If **Create One File** is selected, the generator apart from creating a file per table / class, it will combine the contents of all generated files to one file **merged.txt**.

5 Database Explorer

To open the Database Explorer press the **Database Schema** button in the main window. The Database Explorer lets you create database connections to MS SQL Server and MS Access databases. After creating the connection to the desired database you can navigate the tables and views of this database, view the fields for each table and export (as classes) the selected tables. The Database Explorer lets you export tables from different databases. The Database Explorer window is shown below (**Figure 4 Database Explorer**).

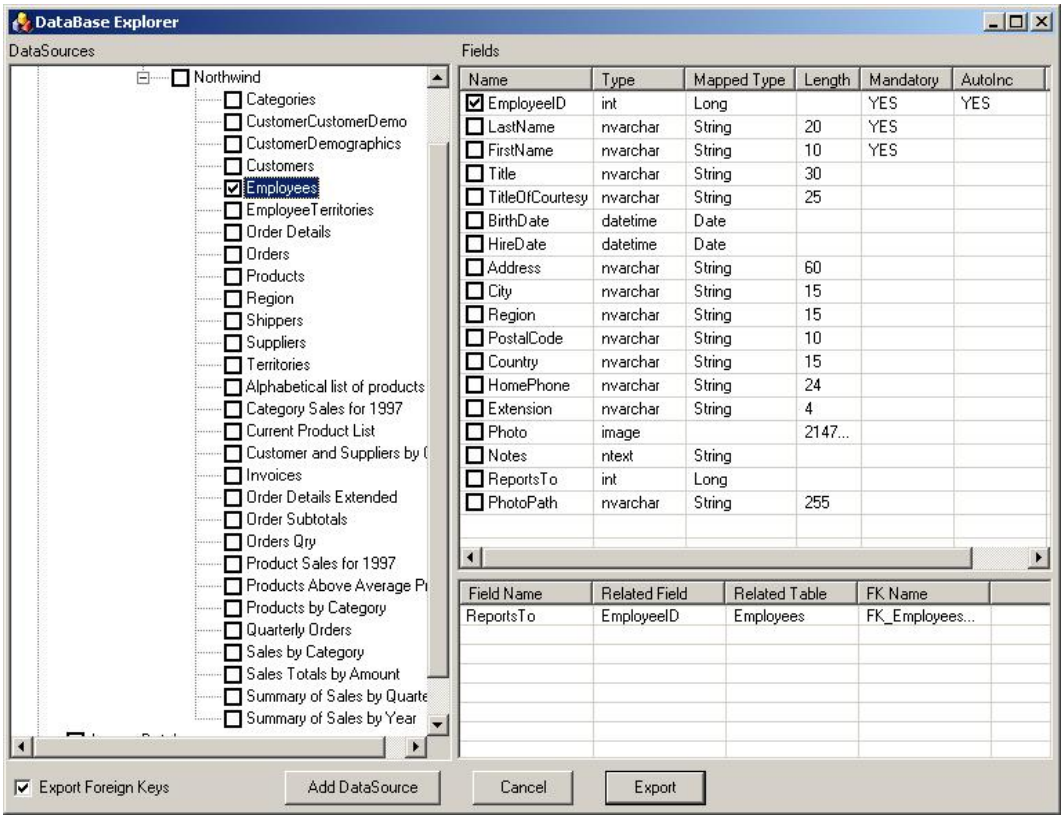

Figure 4 Database Explorer

The Database Explorer is divided in three major sections. On the left lies the database connections tree view. You can see the tables of one database in this pane. On the right we have the fields and the foreign keys list views.

Data sources are added by right-clicking in the connections tree view and selecting the **Add Datasource** menu item, or by pressing the **Add Datasource** button (for details see **5.1 Data sources**).

Each item in the tree view has a check box on the left. You check table items to export to ClassGenerator. By clicking on the item (table) name the fields and foreign keys list views are refreshed. The information one can get from the columns of each list view is explained in the below table:

Table Fields	
Column	**Description**
Checkbox [Primary Key]	If checked, the field is part of the table's primary key
Name	Field name
Type	Field type (SQL Server native types for SQL SQL databases / ADO types for Access databases)
Mapped Type	VB mapped type
Length	Field length (in VB)
Mandatory	True if field is not allowed to be null
Auto Increment	True if the field is auto number (identity)
DB Length	Field length as defined in the database

Foreign Keys	
Column	**Description**

Field Name	The foreign key field name
Related Field	The referenced field name
Related Table	The referenced table name
Foreign Key Name	The foreign key name as defined in the database

The check box **Export Foreign Keys** (default value is checked) is used to export the foreign keys information to ClassGenerator (for more details see **5.2 Class Generator Users Guide**).

You export the selected tables as classes to ClassGenerator by pressing the **Export** button.

5.1 Data sources

You can add new database connections with the **Add Data Source** dialog box (**Figure 5 Add Data Source Dialog Box**).

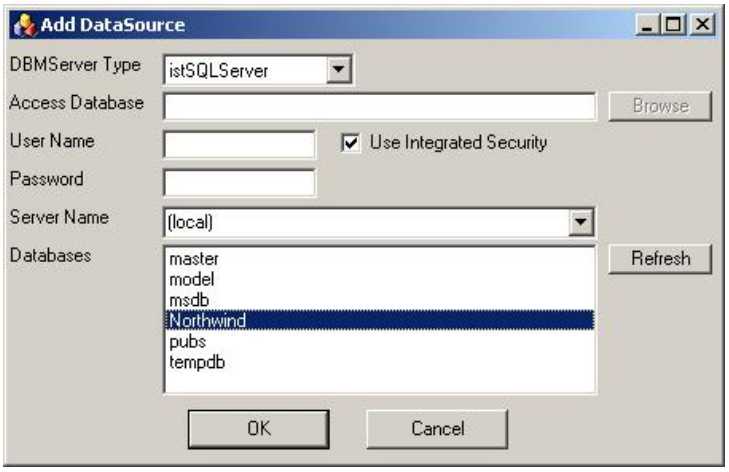

Figure 5 Add Data Source Dialog Box

The fields in the dialog are:

Field	Description
DBMServer Type	Defines the target database MS SQL Server or MS Access.
Access Database	The path of the mdb file. Enabled only when DBMServer Type is Access.
User Name	The user name used to authenticate to the database.
Password	The password for the above user name.
Use Integrated Security	Log in using the windows credentials.
Server Name	The name of the server where SQL Server process runs. Enabled only when DBMServer Type is SQL Server.
Databases	The databases established in the above server. Enabled only when DBMServer Type is SQL Server.

5.2 Export Foreign Keys

If the Export Foreign Keys is checked in the **Database Explorer**, the **Foreign Keys Export** dialog box appears (**Figure 6 Foreign Keys Export Dialog Box**).

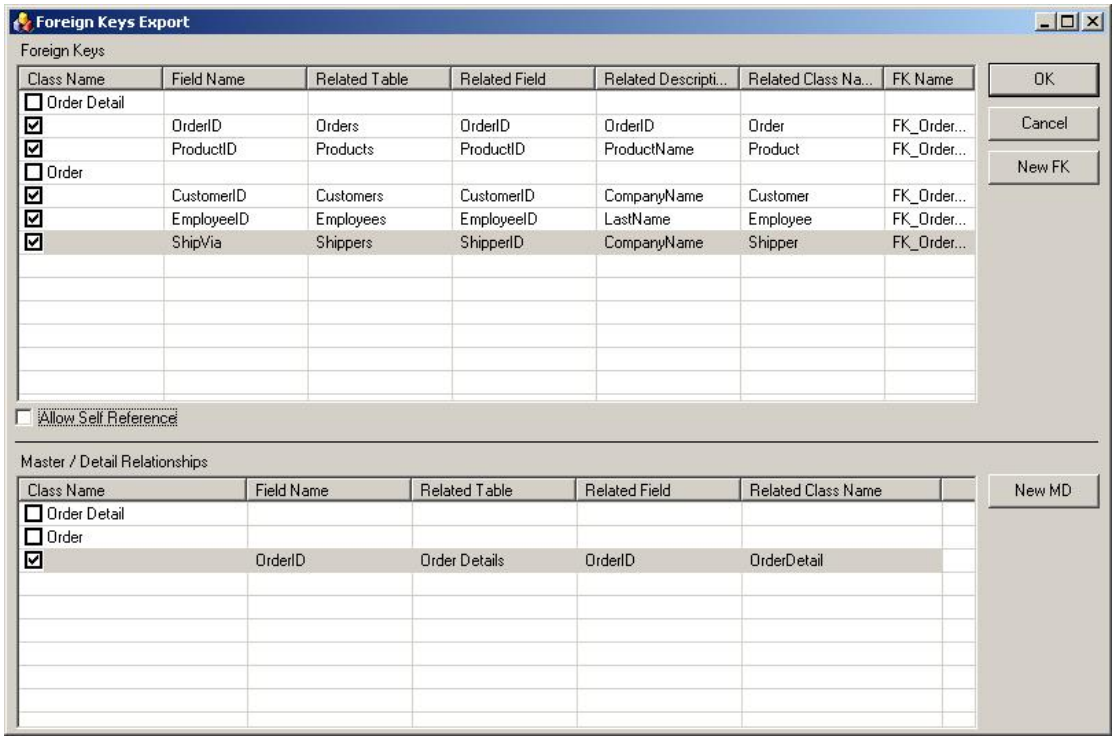

Figure 6 Foreign Keys Export Dialog Box

The fields in the top list view are:

Column	Description
Class Name	The class name for grouping foreign keys data.
Field Name	The foreign key field name
Related Field	The referenced field name
Related Table	The referenced table name
Related Description Field	The description field, which belongs in the Related Table. The data stored in this field are usually shown in GUIs instead of the values in Related Field (e.g. lookup tables).
Related Class Name	The description field, which belongs in the Related Table. The data stored in this field are usually shown in GUIs instead of the values in Related Field (e.g. lookup tables).
Foreign Key Name	The foreign key name as defined in the database

The foreign keys list is already filled-in, if such meta-information is defined in the database. All columns are already filled-in except *Related Description Field* and *Related Class Name*. By right-clicking (context menu) and selecting **Auto Complete Foreigh Keys** the *Related Class Name* is auto completed. Only the *Related Description Field* remains to be filled. You can do that by double-clicking on the row, or by selecting the **Related Description Field** in context menu.

You can cancel editing the foreign key relationship by pressing escape (**ESC**).

The **Allow Self Reference** check-box, is self explanatory, it allows one table to reference itself (and it shows it self in the *Related Table* columns).

In the bottom list view you can create master / detail relationships (e.g. Orders to OrderDetails). These relationships are created manually. You can do that by either selecting the **New MD** button, or the Add Master Detail Relationship in context menu (a table name row must be selected in the list view).

You can cancel editing the master / detail relationship by pressing escape (**ESC**). When editing, you can move from one column to another by pressing **Enter**.

6 Options

In the Options form you can define various application settings, such as:

- Start Tag
- End Tag
- Auto detect tag literals
- Export foreign keys
- Allow self reference

The first three options can be overridden when using template directives.

6.1 Registry Keys / Values

All registry keys used by ClassGenerator reside in:

HKEY_LOCAL_MACHINE\SOFTWARE\ClassGenerator

Data sources are saved in:

HKEY_LOCAL_MACHINE\SOFTWARE\ClassGenerator\DataSources

7 Directives

Directives are special parser commands that can alter its behavior. Only one directive is allowed in each line, starting with two hash marks (**##**).
The complete list of directives follows:

```
##DIRECTIVE START_TAG=<% END_TAG=%>
##DIRECTIVE MAP_TYPE String=String
##DIRECTIVE MAP_TYPE Integer=Integer
##DIRECTIVE MAP_TYPE Long=Integer
##DIRECTIVE MAP_TYPE Double=Double
##DIRECTIVE MAP_TYPE Date=Date
##DIRECTIVE MAP_TYPE Boolean=Boolean
##DIRECTIVE MAP_TYPE Variant=Object
##DIRECTIVE MAP_TYPE Object=Object
##DIRECTIVE MAP_TYPE Decimal=Decimal
##DIRECTIVE FILE_NAME_FORMAT=<%CLASS_NAME%>
```

There are three directive categories:

- start / end tags (##DIRECTIVE START_TAG=<% END_TAG=%>)
- types mapping (##DIRECTIVE MAP_TYPE)
- generated file name format (##DIRECTIVE FILE_NAME_FORMAT)

Note: All directives are case sensitive.

8 Parser Tags

There are 4 types of parser tags:

- Simple / single line tags
- Variable tags (allows base integer arithmetic functions)
- Loops
- If…Then…Else clauses

Note: All tags are case sensitive. One tag must start and end in the same line.

8.1 Simple / single line tags

Tag	Description	Empty Line
CLASS_NAME	The name of the produced class	
TABLE_NAME	The source database table name	
FIELDS_NUMBER	The number of table fields	
PRIMARY_KEY	The field name of the primary key (if the PK is consisted by one field)	
MULTIPLE_PRIMARY_KEYS	The number of fields consisting the PK	
PROJECT_NAME	The name of the project the generated class will be used in	

8.2 Variable tags

Tag	Description	Empty Line
VAR_DEFINE	Defines a variable	TRUE
VAR_WRITE	Outputs variable value to generated file	
VAR_EVAL	Evaluates the new value of the variable using simple functions	TRUE

Examples:

```
<%VAR_DEFINE A = 10%>
<%VAR_DEFINE B = .DAL.%>

<%PROJECT_NAME%><%VAR_WRITE B%><%CLASS_NAME%>

<%FOREACH PROPERTY IN PROPERTIES %>
<% PROPERTY%>
<%VAR_EVAL A = A + 10%>
A = <%VAR_WRITE A%>
<%END FOREACH%>
```

8.3 Loops

Nested **ForEach…In** loops are allowed, without maximum limit. **ForEach…In** loops are allowed in conjunction with **IfThenElse** clauses.

Tag	Description	Empty Line
FOREACH…IN	Start **ForEach…In** loop	
END FOREACH	End **ForEach…In** loop	

Items that can be enumerated by **ForEach…In** statements are:

- PROPERTIES
- PROPERTIES_TYPES
- COUNTER

- PK_FIELDS
- MD_CLASSES_NAMES
- MD_TABLES_NAMES
- MD_PROPERTIES
- MD_PK_FIELDS
- MD_PROPERTIES_TYPES

Property attributes that can be visible in **ForEach…In** statements are:

- CLASS_NAME
- TABLE_NAME
- ITEM_IS_FIRST
- ITEM_IS_LAST
- PROPERTY_IS_AUTO_INC
- PROPERTY_IS_PRIMARY_KEY
- PROPERTY_IS_MANDATORY
- PROPERTY_DB_TYPE
- PROPERTY_LENGTH
- PROPERTY_DB_LENGTH
- PROPERTY_IS_NUMERIC
- PROPERTY_IS_FOREIGN_KEY
- PROPERTY_FK_RELATED_FIELD_NAME
- PROPERTY_FK_RELATED_TABLE_NAME
- PROPERTY_FK_RELATED_DESCR_FIELD_NAME
- PROPERTY_FK_RELATED_CLASS_NAME
- MD_PK_FIELDS_NUMBER
- MD_PROPERTY_IS_AUTO_INC
- MD_PROPERTY_IS_PRIMARY_KEY
- MD_PROPERTY_IS_MANDATORY
- MD_PROPERTY_DB_TYPE
- MD_PROPERTY_LENGTH
- MD_PROPERTY_DB_LENGTH
- MD_PROPERTY_IS_NUMERIC

Note that you have to define these attributes as parameters in **ForEach…In** statement.

All attributes defined in **ForEach…In** are separated with one space (“ ”).

Examples:

```
<%FOREACH PROPERTY PROPERTY_TYPE PROPERTY_IS_AUTO_INC PROPERTY_IS_NUMERIC
PROPERTY_IS_FOREIGN_KEY PROPERTY_IS_MANDATORY IN PROPERTIES PROPERTIES_TYPES
PROPERTY_IS_AUTO_INC PROPERTY_IS_NUMERIC PROPERTY_IS_FOREIGN_KEY
PROPERTY_IS_MANDATORY%>
<%IF PROPERTY_IS_MANDATORY = True AND PROPERTY_IS_NUMERIC = True AND
PROPERTY_IS_AUTO_INC = False THEN%>
        <%PROPERTY%> : Mandatory & Numeric – Not identity field
<%ELSE IF PROPERTY_IS_AUTO_INC = True THEN%>
        <%PROPERTY%> : Identity field
<%END IF%>

<%IF PROPERTY_TYPE = Boolean THEN%>
        <%PROPERTY%> as <%PROPERTY_TYPE%> [boolean]
<%ELSE IF PROPERTY_TYPE = String THEN%>
        <%PROPERTY%>(<%PROPERTY_LENGTH%>) as <%PROPERTY_TYPE%> [string]
<%IF PROPERTY_IS_FOREIGN_KEY = True THEN%>
                PROPERTY_IS_FOREIGN_KEY = True
<%END IF%>
<%ELSE IF PROPERTY_IS_NUMERIC = True THEN%>
        <%PROPERTY%> as <%PROPERTY_TYPE%> [NUMERIC]
<%IF PROPERTY_IS_FOREIGN_KEY = True THEN%>
```

```
                PROPERTY_IS_FOREIGN_KEY = True

        <%PROPERTY_FK_RELATED_TABLE_NAME%>.<%PROPERTY_FK_RELATED_FIELD_NAME%>
[<%PROPERTY_FK_RELATED_DESCR_FIELD_NAME%>]
<%END IF%>
<%END IF%>
<%IF ITEM_IS_LAST = True THEN%>
--------- ITEM_IS_LAST <%PROPERTY%> -------------
<%END IF%>
<%IF ITEM_IS_FIRST = True THEN%>
--------- ITEM_IS_FIRST <%PROPERTY%> -------------
<%END IF%>
<%END FOREACH%>
```

Note: Due to limited page width, lines are wrapped

The result of applying the above template on Northwind tables Orders & OrderDetails follows:

Order.vb

```
        OrderID : Identity field

        OrderID as Long [NUMERIC]
--------- ITEM_IS_FIRST OrderID -------------

        CustomerID(5) as String [string]
                PROPERTY_IS_FOREIGN_KEY = True

        EmployeeID as Long [NUMERIC]
                PROPERTY_IS_FOREIGN_KEY = True
                Employees.EmployeeID [LastName]




        ShipVia as Long [NUMERIC]
                PROPERTY_IS_FOREIGN_KEY = True
                Shippers.ShipperID [CompanyName]

        Freight as Double [NUMERIC]

        ShipName(40) as String [string]

        ShipAddress(60) as String [string]

        ShipCity(15) as String [string]

        ShipRegion(15) as String [string]

        ShipPostalCode(10) as String [string]

        ShipCountry(15) as String [string]
--------- ITEM_IS_LAST ShipCountry -------------
```

OrderDetail.vb

```
        OrderID : Mandatory & Numeric – Not identity field

        OrderID as Long [NUMERIC]
                PROPERTY_IS_FOREIGN_KEY = True
                Orders.OrderID [OrderID]
--------- ITEM_IS_FIRST OrderID -------------
        ProductID : Mandatory & Numeric – Not identity field

        ProductID as Long [NUMERIC]
                PROPERTY_IS_FOREIGN_KEY = True
                Products.ProductID [ProductName]
        UnitPrice : Mandatory & Numeric – Not identity field

        UnitPrice as Double [NUMERIC]
```

```
        Quantity : Mandatory & Numeric – Not identity field

        Quantity as Integer [NUMERIC]
        Discount : Mandatory & Numeric – Not identity field

        Discount as Double [NUMERIC]
--------- ITEM_IS_LAST Discount ------------
```

8.4 If…Then…Else clauses

Nested **If…Then…Else** clauses are allowed in conjunction with **ForEach** loops (no maximum limit of nested statements exists).

Tag	Description	Empty Line
IF … THEN	Start **If…Then** clause	
ELSE	Start **Else** clause	
ELSE IF … THEN	Start **Else If…Then** clause	
END IF	End **If** clause	

Most of property attributes are allowed to be used in **If…Then…Else** clauses (examples are shown in **ForEach…In** section):

- MULTIPLE_PRIMARY_KEYS
- PK_FIELDS_NUMBER
- PRIMARY_KEY_IS_AUTO_INC
- PROPERTY_IS_AUTO_INC
- PROPERTY_IS_PRIMARY_KEY
- PROPERTY_IS_MANDATORY
- PROPERTY_DB_TYPE
- PROPERTY_LENGTH
- PROPERTY_DB_LENGTH
- PROPERTY_IS_NUMERIC
- PROPERTY_IS_FOREIGN_KEY
- PROPERTY_FK_RELATED_FIELD_NAME
- PROPERTY_FK_RELATED_TABLE_NAME
- PROPERTY_FK_RELATED_DESCR_FIELD_NAME
- ITEM_IS_LAST
- ITEM_IS_FIRST
- HAS_MASTER_DETAIL_CLASSES
- MD_PK_FIELDS_NUMBER
- MD_PROPERTY_IS_AUTO_INC
- MD_PROPERTY_IS_PRIMARY_KEY
- MD_PROPERTY_IS_MANDATORY
- MD_PROPERTY_DB_TYPE
- MD_PROPERTY_LENGTH
- MD_PROPERTY_DB_LENGTH
- MD_PROPERTY_IS_NUMERIC