SQL Server 2005 - SQL Server Integration Services – Parte 5 - Loop Containers

As we briefly mentioned in the previous installment of this series, among features introduced in the SQL Server 2005 Integration Services there are For and ForEach loops, implemented in the form of containers that can be incorporated into the Control Flow part of a package design. (They are included in the list of package components listed in the Toolbox accessible from the Control Flow tab of the SSIS Designer interface of the SQL Server Business Intelligence Development Studio). Both of them serve the same generic purpose (familiar to anyone with even minimal exposure to programming languages), which is repetitive execution of a sequence of tasks, as long as an arbitrarily chosen condition is satisfied (status of the condition is checked at the end of each execution sequence). In the SQL Server 2000 (and 7.0) Data Transformation Services, such functionality was not available and emulating it required rather cumbersome workarounds.

In the case of the For Loop container, the decision of whether execution of components within its scope should be repeated is based on the value of three interrelated expressions:

- EvalExpression - a Boolean (one that evaluates to either True or False) expression that is used to determine whether the sequence of tasks within the container should be executed or whether the flow of control should be transferred to the next container in the package (or be terminated, if the For Loop happens to be the last container within the package).
- InitExpression - typically, an assignment (assigning a value to a variable) expression that impacts the result of evaluation of EvalExpression the first time control flow enters the loop. If the evaluation yields a False result, the control flow exits the loop (in other words, no tasks within the For Loop are executed).
- AssignExpression - also, most commonly, an assignment expression, which is used to update values in EvalExpression at the end of each iteration. Once values are updated, evaluation takes place again.

All three properties must follow the syntax of SSIS expressions, which we discussed in our previous article. The only deviation from this rule is the use of assignment operator, which is not allowed in any SSIS context other than the properties of the For Loop container. Expression in this case takes the form `@variable = expression`, where the `expression` must result in a data type compatible with the data type of the variable to which its value is assigned. All three properties can be set from the For Loop Editor interface or from the Properties window (note, however, that the Expression Builder interface is not available when performing this task). Within the For Loop Editor, it is possible to assign to properties both direct values as well as expressions (which we discussed earlier in this series of articles).

We will demonstrate the For Loop functionality through a very simple package, which, besides the For Loop container itself, also includes a single Script Task (within the container scope). We will also use a single variable iCounter of data type Int16 as the loop counter, serving as an element of InitExpression, EvalExpression, and AssignExpression expressions. To accomplish this, create a new (or use an existing) SSIS project within the SQL Server Business Intelligence Development Studio. Ensure that the Control Flow tab is selected and drag the For Loop Container icon from the Toolbox to the Designer area. With the newly created container selected, display the Variable window (this can be done from the View menu, pointing to the Other Windows submenu and clicking on the Variables entry).

Clicking on the first icon in the Variables window toolbar will create a variable with the scope of the For Loop Container. In the Name column, type in iCounter and choose the Int16 from the drop-down list as the Data Type. Right-click on the For Loop Container and select Edit from the context-sensitive menu. Fill out the For Loop Properties section of the For Loop Editor according to the following list:

- for the expression `InitExpression` type in `@iCounter=0`
- for the expression `EvalExpression` type in `@iCounter<3`
- for the expression `AssignExpression` type in `@iCounter=@iCounter+1`

Click on OK to close the For Loop Editor. Next, drag a Script Task component from the Toolbox and drop it inside the For Loop container. Right-click on it and select the Edit item from the context-sensitive menu. You will be presented with the Script Task Editor interface, which is fairly straightforward. The portion that is relevant to us, accessible by clicking on the Script entry on the left hand side of the window, allows you to:

- set the programming language (note that the term "script" used in this case is a bit of misnomer), which in our case will be Microsoft Visual Basic .NET, specify precompilation options (PrecompileScriptIntoBinaryCode),
- define an entry point where execution will start (through the EntryPoint property, which, by default has the value of ScriptMain), and
- specify SSIS variables, which can be only read (ReadOnlyVariables) or read and written to (ReadWriteVariables) within the code.

In our case, the only change to default settings will be adding iCounter to the ReadOnlyVariables box. Actual code is accessible by clicking on the Design Script... button in the lower right corner of the Script Task Editor. This launches Microsoft Visual Studio for Applications, where you will notice the ScriptMain window containing the public class with the same name and its only Sub called Main. Within the code for Sub Main(), you should see a commented out entry stating `"Add your code here"` followed by the line of code `Dts.TaskResult = Dts.Results.Success`. We will add a single line in between these two, containing `MsgBox(Dts.Variables("iCounter").Value.ToString)` (the purpose of this code is to simply display the value of the iCounter User type variable in a message box, which will indicate changes to it on every iteration of the control flow through the For Loop container). Once this is done, your entire Main Sub should resemble the code below:

```
Public Sub Main()
        '
        ' Add your code here
        '
        MsgBox(Dts.Variables("iCounter").Value.ToString)
        Dts.TaskResult = Dts.Results.Success
End Sub
```

Close the Visual Studio for Applications window and click on the OK button in the Script Task Editor to return to the Designer interface. If you select the For Loop container and right-click on it, you will notice the Execute Task option in its context-sensitive menu (this allows you to launch individual tasks, without executing the entire package). Selecting it will result in the For Loop Container and the Script Task within it turning yellow, and shortly afterwards, a message box with the Script Task header, value 0, and OK button being displayed. After clicking on the OK button, you should see another message box with a value of 1, followed (after clicking on OK again) with another one with a value of 2. Confirming the

message again will complete the execution of the For Loop (as expected, since the iCounter reached the value of 3, causing the EvalExpression to evaluate to False), which is indicated by the For Loop Container and Script Task changing their color from yellow to green. To stop execution, select the Stop Debugging from the Debug menu (or press the Shift + F5 key combination).

While For Loop container determines the number of iterations by checking whether an arbitrary condition evaluates to True or False, the ForEach loop derives this result through enumeration. Enumeration can be applied to a number of different collections, dependent on enumerator types:

- Foreach File Enumerator - evaluates the number of iterations based on the number of files residing in the same folder. It is possible to specify additional conditions that will further refine your selection, such as inclusion of subfolders or filtering based on the file extension or match on any part of file name. Standard wildcard characters, such as the asterisk (designating any combination of characters) and the question mark (designating a single character) can also be used.
- Foreach Item Enumerator - allows you to specify explicitly the items belonging to the collection to be enumerated.
- Foreach ADO Enumerator - is used for enumeration of rows in a recordset referenced by an ADO object variable.
- Foreach ADO.NET Schema Rowset Enumerator - works with schema information about a data source (accessible via OLE DB connection), such as Tables, Views, Columns, Collations, Foreign Indexes, etc.
- Foreach From Variable Enumerator - applies enumeration to the content of a variable, which represents an object or collection that is enumerable.
- Foreach NodeList Enumerator - works by applying XPath syntax to retrieve specific nodes from the result set generated by an XML Path Language expression.
- Foreach SMO (SQL Server Management Object) Enumerator - retrieves instances of a particular type of a SQL Server Management object on the level of the server (covering such object types as Linked Servers, Jobs, or Logins), database, (including File Groups, Data Files, Log Files, Stored Procedures, User Defined Data Types, User Defined Functions, Users, Views, and Tables), or table (Columns, Foreign Keys, Triggers).

We will provide another very straightforward example illustrating the operation of the Foreach Loop container. For the sake of simplicity, we will limit our focus in this article to Foreach File Enumerator, however we will be discussing the remaining ones in more detail in future articles. In order to test its functionality, you can take advantage of the same package that was used earlier when discussing the For Loop container. Start by dragging the Foreach Loop container icon from the Toolbox onto the Designer area of the SQL Server Business Intelligence Studio. With the container selected, create a new variable called sFileName of String datatype and Foreach Loop Container scope. Create a new Script Task (as before), but this time type in the sFileName as the ReadOnlyVariables entry. Modify the Sub Main() of the script so it contains the following code:

```
Public Sub Main()
        '
        ' Add your code here
        '
        MsgBox(Dts.Variables("sFileName").Value.ToString)
        Dts.TaskResult = Dts.Results.Success
End Sub
```

Next, in the Foreach Loop Editor interface (which you can display by selecting Edit item from the context-sensitive menu of the Foreach Loop container), click on the Collection item (appearing on the left hand side of the Editor window). Set the Enumerator type to Foreach File Enumerator (by choosing the appropriate entry from the drop down list) and specify the name of a Folder or locate it after clicking on the Browse... button (this can be any of the folders on a local or remote computer that you have at least read access to). Provide the name of the files within this folder that you want to work with, and pick one of three options that control the retrieved file name (fully qualified, name only, or name and extension).

Finally, in the Foreach Loop Editor interface, click on Variable Mappings, and set the User::sFileName variable to Index 0. This will associate our variable with the first column (since the index is zero-based) in the enumerated collection, which contains names of files that match criteria defined on the Collection page.

Click on OK to confirm your choices and execute the content of the container by highlighting it and using the already familiar "Execute task" item from the context sensitive menu. You should see the names of the files from the target folder, satisfying the filtering condition that you specified, being displayed one by one in the dialog boxes similar to the ones we have seen when experimenting with the For Loop container.

In our next article, we will look closer into the remaining types of Foreach Loop containers.