SQL Server 2005 - SQL Server Integration Services - Part 8 - Foreach Loop Container

We are continuing our overview of different types of enumerators available in the Foreach Loop Container of SQL Server 2005 Integration Services. The ones covered so far provided the ability to iterate over a set of files matching an arbitrary set of criteria (Foreach File Enumerator), records within an ADO recordset (Foreach ADO Enumerator), a collection of SQL Server Management objects (Foreach SMO Enumerator), or components of XML-formatted data (Foreach Node Enumerator). The next two enumerators in this series - Foreach Item and Foreach From Variable - further extend the scope of data processing capabilities integrated into the Foreach Loop Container.

While the Foreach Item enumerator has rather limited use, in some scenarios, its simplicity, convenience, and independence from external data sources, make it an option worth considering. In particular, it works well in situations dealing with arbitrarily selected, relatively static (as far as frequency of changes are concerned) set of items that need to be processed repetitively in an identical manner and that are not easily grouped using other types of enumerators (such items might include database objects, which do not share common characteristics, computer or user names that designate data source or destination, etc.). While you can easily store this type of data in a database table or a text file (and retrieve it prior to initiating actions in the loop), placing it within the loop itself (as part of the properties of its enumerator) might be preferred, if you want to make a package self-contained or at least decrease the level of its independency on external data stores.

In order to get familiar with basic characteristics of the Foreach Item enumerator, let's walk through the generic steps involved in its setup. As usual, start by creating a new SQL Server Integration Services project using SQL Server Business Intelligence Development Studio, and drag the Foreach Loop Container icon from the Toolbox menu to the Control Flow tab of the Designer interface. Right-click on it and select Edit item from the context sensitive menu. In the Foreach Loop Editor, select the Collection entry. Pick the Foreach Item Enumerator from the drop-down list to the right of the Enumerator label. This will reveal the Enumerator configuration section in the area below, allowing you to specify individual items that will be enumerated. This requires defining the number of columns that will contain data about enumerated items as well as assigning a data type for each. Once this is completed, you simply type values in each column for every item on the list. Finally, in the Variable Mappings section of the Foreach Loop Editor, declare variables that will be mapped to individual columns (i.e. Column 0 corresponds to a variable with Index 0, Column 1 corresponds to a variable with Index 1, etc.). This way, you can initiate actions within the loop based on values of one or more properties of enumerated items.

The Foreach From Variable enumerator, similiar to the Foreach Item enumerator, is likely to trail other types of enumerators in terms of popularity. Its main purpose is providing the ability to enumerate elements of a collection that has been created outside of the Foreach Loop container. You can accomplish this by defining a package-scope SSIS variable of the Object data type and populating it during package execution (for example, by executing a Script Task). The following sample will demonstrate this approach by enumerating all of the local SQL Server databases, which share a common owner (you can easily change the qualifying criteria by applying simple modifications to the script used in our example).

Start by creating a new SQL Server Integration Services project. As you might expect, the primary prerequisite, which we need to take care of before proceeding further, is the creation of a variable that will hold the collection to be enumerated. To accomplish this, activate the Variables window (from View -> Other Windows menu) and add a new variable (which we will call colDB_SSIS) of the object type and package scope. Next, drag the Script task icon from Toolbox to the Control Flow tab of the Designer interface, right click on it, and select Edit from its context sensitive menu. Switch to the Script section of the Script Task Editor, add colDB_SSIS as the ReadWriteVariables entry and click on the Design Script... command button. This will display the Microsoft Visual Studio for Applications window. Since we are planning to use variables of type Database in our code, we need to add reference to the Microsoft.SqlServer.ConnectionInfo .NET component. This can be done either from the context sensitive menu of Reference node in the Project Explorer window or via the Add Reference... item in the Project menu (in either case, you will be presented with the Add Reference dialog box where required components can be located and added). Next, type in the new code within the Public Sub Main() so it resembles the following:

```
Imports Microsoft.SqlServer.Management.Smo
Public Sub Main()
    '
    ' Add your code here
    '
    Dim oSrv As Server = New Server()
    Dim colDB As Collection = New Collection()
    Dim oDB As Database
    For Each oDB In oSrv.Databases()
        If oDB.Owner.Equals("Owner") Then
            colDB.Add(oDB)
            MsgBox(oDB.Name & vbTab & colDB.Count)
        End If
    Next
    Dts.Variables("colDB_SSIS").Value=CObj(colDB)
    Dts.TaskResult = Dts.Results.Success
End Sub
```

The code above uses two variables of Server and Database class types. In the For loop, we iterate through the list of all databases hosted on the local default instance of SQL Server, searching for the ones with the owner matching our arbitrary value (note that you should replace the *"Owner"* entry with one that would be valid in your environment - for testing purposes, you can simply use "sa", which, at the very least, should result in creating a collection of all the system databases). For every match, we add the current value of the Database object to the collection. We also display a message box containing the name of the database and total number of databases in our collection. Once the processing is completed, we assign the content of the collection to the value of the colDB_SSIS package level variable.

After you finish copying the code to the Public Sub Main(), close the Microsoft Visual Studio for Applications and Script Task Editor windows and return to the Designer interface. Drag the Foreach Loop Container icon from the Toolbox menu to the Control Flow tab of the Designer interface placing it directly underneath Script Task, right-click on it, and select the Edit item from the context sensitive menu. In the Foreach Loop Editor, activate the Collection entry. Make sure that the Foreach From Variable Enumerator appears in the Enumerator drop-down list, and then choose User::colDB_SSIS as the Variable in the Enumerator configuration section. Switch to the Variable Mappings section. In the Variable Column, use entry to create a variable (we will call it oDB) of the Object type and Foreach Loop Container scope (this will automatically associate it with Index 0). Close the Foreach Loop Editor window and extend Precedence Constraint (green arrow) from the bottom of the Script Task to the top of the Foreach Loop Container.

In order to verify that the package works as intended, we will add a Script Task to the Foreach Loop Container. Its sole purpose in this example is to display the name of the currently processed database throughout each loop iteration. Drag its icon from the Toolbox into the rectangle representing the Foreach Loop Container and activate its Editor window (using the Edit... item in its context sensitive menu). Switch to the Script section, type in oDB in the ReadOnlyVariables entry, and click on the Design Script... command button. Add a single line of code to its Public Sub Main() so it looks as follows:

```
Public Sub Main()
    '
    ' Add your code here
    '
    MsgBox(Dts.Variables("oDB").Value.ToString)

    Dts.TaskResult = Dts.Results.Success
End Sub
```

Close the Microsoft Visual Studio for Applications and Script Task Editor windows, and execute the package. You should see the sequence of dialog boxes generated by the first Script Task, containing names of databases with the owner you specified in place of the *"Owner"* entry, along with the increasing collection count. This should be followed by the identical sequence (this time without the collection count), generated by the second Script Task (within the Foreach Loop with the From Variable enumerator).