SQL Server 2005 - SQL Server Integration Services - Part 5 - Foreach ADO

In the previous article of this series, we started reviewing two new types of containers introduced in the SQL Server 2005 Integration Services - For Loop and Foreach Loop. We also provided a simple example illustrating the use of Foreach File enumerator. We will continue our review of Foreach enumerators starting with Foreach ADO, since this one probably qualifies as the most popular choice in its category. This is due to the fact that ADO recordsets offer a convenient way of dealing with data sources, regardless of their type (for example, within SSIS, you can easily populate a recordset by executing a SQL query or by reading the content of a flat file). The goal of this article is to demonstrate the second of these scenarios and, at the same time, explain how to retrieve data from flat files processed within the Foreach Loop with File Enumerator (building on the information provided previously). To accomplish this, we will place Foreach Loop with ADO Enumerator within the Foreach Loop with File Enumerator (such arrangement is referred to as loop nesting). For the sake of simplicity, we will divide the process into two stages. First, we will show how to store the content of a text file in a recordset and process it in the Foreach Loop based on the ADO enumerator. Next, we will modify the package by enclosing this loop within another one, based on File Enumerator.

Let's start by creating a delimited text file, which will serve as a source of data that will populate our recordset. We choose the semicolon as a column separator, with each row occupying a separate line. Our sample file looks as follows:

```
Thievery Corporation;The Mirror Conspiracy
Massive Attack;The 100th Window
Zero 7;simple life
```

Save the file as "DesertIsland1.txt" in the C:\DatabaseJournal\SSIS folder (obviously choices of the content, file name, and its location are completely arbitrary). Launch the SQL Server 2005 Business Intelligence Development Studio and create a new Integration Services Project. You can also use the one we created for the purpose of demonstrating functionality of Foreach Loop with File Enumerator. (As we mentioned before, it is possible to execute individual tasks or containers by selecting Execute Task from their context sensitive menu - alternatively, you can also disable individual ones that are not needed - which eliminates the need to delete them or create a new package). As you might remember, in order to access the content of a text file, we need to have an appropriate connection manager. Right-click on the Connection Managers area of the Package Designer interface and select the "New Flat File Connection..." menu item. Type in "Desert Island" as the Connection manager name, point to the newly created file using the Browse... button next to the File name label, and accept the default choices in the General section of the Flat File Connection Manager Editor. Confirm that options for row and column delimiters are set correctly in the Columns section, ensure that both Column 0 and Column 1 are listed as string data type in the Advanced section, and take a quick look at the Preview for final verification.

Now it is time to take care of the creation of a data source and resulting recordset. Both of them are part of the Data Flow portion of the package implementation - which means that you need to either drop Data Flow task from the Toolbox onto the Control Flow design area and double click on it or switch to the Data Flow tab of the Designer interface and create a new Data Flow task from there. In either case, you will end up with an empty Data Flow area representing the newly created task. Drag the Flat File Source entry from the Toolbox onto it, right-click on it, and select

Edit from the context-sensitive menu. This will display the Flat File Source Editor, with "Desert Island" appearing in the Flat file connection manager drop down list (it is selected automatically since this is the only object of this type in our package). Clicking on the Preview button appearing below will display the content of the "DesertIsland1.txt" file in the two-column format. Click on OK to close the Editor window. Next, drag the Recordset Destination entry from the Toolbox onto the Data Flow area and position it directly below the Flat File Source. Click on the rectangle representing Flat File Source to make it active and drag the green arrow from the middle of its bottom edge all the way to the top edge of the Recordset Destination (you can use items in the Format menu in order to align them according to your preferences).

In order to be able to access the recordset outside of the Data Flow task, we will create a Package-level variable called rsFileContent of data type Object. To accomplish this, switch to the Control Flow tab, click on the empty area of the Designer interface (so none of its containers or tasks are selected), activate the Variables window (from View or SSIS) menu, and click on the New Variable icon in its toolbar. Type rsFileContent as the variable name, ensure that Package appears in the scope column, and select Object in the Data Type column. Close the Variables window and switch back to the Data Flow tab of the Designer interface.

Display the Advanced Editor for Recordset Destination by choosing the Edit entry in its context sensitive menu. In the VariableName entry of Custom Property section on the Component Properties tab, type in rsFileContent. Confirm that Column 0 and Column 1 appear under the Input Columns tab and mark the checkboxes next to each of them. Accept the remaining defaults and click on OK to close the Advanced Editor window.

When the package is executed, the content of the text file populates the rsFileContent variable. In order to retrieve individual rows, we will use Foreach Loop with ADO Enumerator. Click on the Control Flow tab of the Designer interface to activate it and drag the Foreach Loop Container icon from the Toolbox to the Designer area directly underneath the previously created Data Flow task. Extend the green arrow, visible when the Data Flow task is selected, to the top edge of the Foreach Loop container and select Edit from its context-sensitive menu to display the Foreach Loop Editor. In its Collection section, select Foreach ADO Enumerator in the Enumerator entry, choose User::rsFileContent from the drop down list, which designates the ADO object source variable, and keep the default "Rows in the first table" selection for the Enumeration mode. Since our recordset contains two columns, we will need two variables to store the content of each for every row. You can create them directly from the Variable Mappings section of the Foreach Loop Editor - simply by selecting the <New Variable...> option in the Variable column. In our case, we will define Package level variables sColumn0 and sColumn1 for storing entries from Column 0 and Column 1, respectively (which are represented by Index 0 and 1 in the Foreach loop). In the Add Variable dialog box, select Package as the scope and String as the data type from drop down lists and type in an appropriate variable Name. Once User::sColumn0 is assigned index 0 and User::sColumn1 is assigned to index 1, click on OK to close the Foreach Loop Editor window and return to the Designer area.

To verify that we accomplished our goal, we will display message boxes with the content of each row, for every loop iteration. For this, we will need to have a Script task, similar to the one we have used in our previous examples. To create it, drag its icon from the Toolbox and drop it inside the Foreach Loop Container. Right click on it and select Edit from the context-sensitive menu. In the Script section of the Script Task Editor, in the ReadOnlyVariables, type in sColumn0,sColumn1. Click on

the Design Script... button and modify the Public Sub Main() in the Visual Studio for Applications window so it looks as follows:

```
Public Sub Main()
 '
 ' Add your code here
 '
 MsgBox(Trim(Dts.Variables("User::sColumn0").Value.ToString) & " --- " & _
  Trim(Dts.Variables("User::sColumn1").Value.ToString))
 Dts.TaskResult = Dts.Results.Success
End Sub
```

This will display the content of both fields from each row of the recordset on every iteration of the Foreach loop. Trim function, which we included above, simply removes trailing (and potentially leading) spaces, making the display a bit more compact. Once you execute the package, you should see three message boxes, displayed in sequence, each containing an entry from each column of every row in the recordset (originating from our DesertIsland1.txt text file).

By putting together a package that displays the content of recordset populated from a text file, we accomplished the first of our goals. Now let's try to extend this functionality by processing multiple files in repetitive fashion (each of them will produce a separate recordset that will be processed using the same set of components that we just created). Since we will need to iterate over a collection of files, we need another Foreach Loop (this time with File Enumerator). Drag its icon from the Toolbox onto the Control Flow area of the Designer interface and position it to the side of our Data Flow task and Foreach Loop container. Next, select both of these components and drag them over the newly added Foreach Loop container, which will make them part of its content. Right-click on the outer Foreach Loop and select Edit from the context sensitive menu. In the resulting Foreach Loop Editor, ensure that Foreach File Enumerator appears in the Enumerator entry of the Collection section, point to C:\DatabaseJournal\SSIS in the Folder entry in the Enumerator configuration area, type in DesertIsland*.txt in the Files text box, and keep the default "Fully qualified" entry in the "Retrieve file name" area. Switch to the Variable Mappings section, define a new variable with the scope set to the current Foreach Loop, with Name sDesertIslandFile, String value type, and Index of 0 (these steps should be familiar from our previous article).

As before, in order to track the progress of the package execution, we will add the Script Task displaying a message containing the name of the currently processed file. To accomplish this, place a new Script Task inside the outer Foreach Loop Container, above the Data Flow task. Extend the green arrow from the bottom of the Script task to the Data Flow task. Finally, display the Editor window of the Script Task (using the Edit item from its context-sensitive menu), switch to the Script section, add sDesertIslandFile in the ReadOnlyVariable section, click on the Design Script... button, and modify the Public Sub Main() so it has the following content:

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

To properly test our package, add another text file called, for example, "DesertIsland2.txt" in the C:\DatabaseJournal\SSIS folder (or whatever naming convention and location you decided on) and include a few entries within it, following the same format as the one used in "DesertIsland1.txt", for example:

VAST;Visual Audio Sensory Theater
Dead Can Dance;A Passage in Time
Lisa Gerrard;Duality

In the last remaining step, we need to adjust our package to reflect the fact that the Flat File Connection Manager should be pointing to each file retrieved in the outer Foreach Loop (the one using the File Enumerator). Otherwise, this loop will execute twice (since we have two files in the target folder that match our criteria), but both times it will display the content of the "DesertIsland1.txt" - since this is the name of the file specified in the "Desert Island" connection manager. To change this, select the "Desert Island" connection manager icon and, in its Properties window, find the Expressions entry. Next, click on the ellipsis (...) button on its right-hand side. In the Property Expression Editor, choose ConnectionString from the drop down list in the Property column and then assign @[User::sDesertIslandFile] to it (you can use the Expression Builder to avoid typing mistakes), since this is where names of files retrieved in the outer Foreach Loop are stored. Once the package executes, it will process both files, displaying their names and the content of each.

In our next article, we will continue our review of the Foreach Loop container enumerators in the SQL Server 2005 Integration Services.