http://www.sqlservercentral.com/articles/XML/62054/
Printed 2008/03/13 06:35PM

XML Workshop XV - Accessing FOR XML results with ADO.NET

By Jacob Sebastian, 2008/01/23

Introduction

In a few of the early sessions of the XML Workshop, we have seen how to generate XML results from TSQL. We have examined the FOR XML clause and have seen the usages of RAW, AUTO, PATH and EXPLICIT.

XML Workshop 1 - Explains FOR XML with AUTO and RAW

XML Workshop 3 - Explains FOR XML with PATH

XML Workshop 4 - Explains FOR XML with EXPLICIT

Accessing results of FOR XML from a .NET application

Most of the times the results of FOR XML queries are expected to be consumed by client applications. In this session, let us see how to access the results of FOR XML queries from VB.NET and C#.NET applications using ADO.NET.

Sample Table and Stored Procedure

Let us create a sample table and populate it with some data. Let us then create a stored procedure which generates an XML document using FOR XML.

```
-- Let us create a new database
CREATE DATABASE XmlTest
GO

-- Let us now create a table for our example
USE XmlTest
GO

CREATE TABLE Employee (EmpID INT IDENTITY(1,1), EmpName VARCHAR(50) )
GO

-- let us insert some data
INSERT INTO Employee (EmpName)
SELECT 'Jacob' UNION ALL
SELECT 'Michael' UNION ALL
SELECT 'Richard'
```

Let us now create a stored procedure which generates the XML document that we need.

```
CREATE PROCEDURE GetEmployeeData
AS
```

```
SELECT * FROM Employee
FOR XML AUTO, ROOT('Employees')
```

Here is the result of the stored procedure.

```
<Employees>
  <Employee EmpID="1" EmpName="Jacob" />
  <Employee EmpID="2" EmpName="Michael" />
  <Employee EmpID="3" EmpName="Richard" />
</Employees>
```

VB.NET Sample Code

Let us create a VB.NET console application which executes the above stored procedure and fetches the XML document. Here is the code which does that.

```
' references
Imports System.Data.SqlClient
Imports System.Xml
Imports System.Text
Module Module1

    Sub Main()
        'let us make a connection first
        Dim str As String
        str = "Data Source=TOSHIBA-USER\SQL2005;Initial Catalog=XmlTest;"
        str = str + "Persist Security Info=True;User ID=sa;Password=sa2005"
        Dim cn As New SqlConnection(str)
        cn.Open()

        'Let us make a command
        Dim cmd As New SqlCommand()
        cmd.Connection = cn
        cmd.CommandText = "GetEmployeeData"
        cmd.CommandType = CommandType.StoredProcedure

        'What we need next is an XMLReader and call
        'ExecuteXMLReader method of SqlCommand.
        Dim r As XmlReader
        r = cmd.ExecuteXmlReader()

        'Read the data from XMLReader and Load into
        'the String Builder
        Dim xmlData As New StringBuilder
        Do While r.Read()
            xmlData.Append(r.ReadOuterXml())
        Loop

        'Print the output
        Console.WriteLine(xmlData.ToString)

        'Close the Reader and DB Connection
        r.Close()
        cn.Close()

    End Sub

End Module
```

C#.NET Sample Code

Now, let us write the C#.NET version of the above code.

```
using System;
using System.Data.SqlClient;
using System.Data;
using System.Xml;
using System.Text;

namespace ConsoleApplication2
{
    class Program
    {
        static void Main(string[] args)
        {

            //let us make a connection first
            String str;
            str = "Data Source=TOSHIBA-USER\\SQL2005;Initial Catalog=XmlTest;";
            str = str + "Persist Security Info=True;User ID=sa;Password=sa2005";
            SqlConnection cn = new SqlConnection(str);
            cn.Open();

            //Let us make a command
            SqlCommand cmd = new SqlCommand();
            cmd.Connection = cn;
            cmd.CommandText = "GetEmployeeData";
            cmd.CommandType = CommandType.StoredProcedure;

            //What we need next is an XMLReader and call
            //ExecuteXMLReader method of SqlCommand.
            XmlReader r;
            r = cmd.ExecuteXmlReader();

            //Read the data from XMLReader and Load into
            //the String Builder
            StringBuilder xmlData = new StringBuilder();
            while (r.Read())
            {
                xmlData.Append(r.ReadOuterXml());
            }

            //Print the output
            Console.WriteLine(xmlData.ToString());

            //Close the Reader and DB Connection
            r.Close();
            cn.Close();

        }
    }
}
```

Conclusions

This article presents an example of accessing FOR XML results from a .NET application. I guess there must be other ways of doing this too. This sample code is created for the purpose of demonstrating the basic usage. The sample applications are tested with the sample data and found to be working correctly. However, please note that the code is not highly optimized. You might need slightly different code for a production level application. For example, you might need to dispose() the database connection after reading the information. I leave that to the .NET developer in you.
