Passing an Array of Items to a SQL Stored Procedure Using XML from C#

In SQL Server, a common challenge is passing multiple values to a stored procedure. One effective way to achieve this is by passing an array or list of items as a parameter allowing you to compactly represent an array of items as a single parameter, making it easy to process complex datasets without numerous calls to the database.

Step 1: Define the SQL Stored Procedure

Create a stored procedure that accepts an XML parameter and use the nodes method to traverse the XML and extract items into a table variable for further processing.

```
use [YourDatabase]
go

drop procedure if exists [dbo].[ProcessItemsFromXML]
go

create procedure [dbo].[ProcessItemsFromXML]
  @Items XML
as
begin
    -- declare a table variable to store the XML parsed items
    declare @itemtable table (Item nvarchar(50));

    -- parse the XML and insert items into the @itemtable table variable
    insert into @itemtable (Item)
    select x.value('.', 'nvarchar(50)')
    from @Items.nodes('/Items/Item') as t(x);

    -- select the items
    select * from [dbo].[YourTable] where [YourColumn] in (select [Item] from
@itemtable);
end
go
```

Step 2: Create the C# method to pass XML to the Stored Procedure

Write a C# method that constructs the XML from an array of items and calls the stored procedure using "XElement" from the "System.Xml.Linq" namespace to create the XML structure.

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

public class DataProcessor
{
  public void ExecuteStoredProcedureWithXML(SqlConnection connection,
string[] items)
    {
      // create test XML structure
      var xmlItems = new XElement("Items",
          items.Select(item => new XElement("Item", item)));

      using (SqlCommand command = new SqlCommand("ProcessItemsFromXML",
connection))
      {
        command.CommandType = CommandType.StoredProcedure;

        // add the XML parameter
        SqlParameter xmlParam = command.Parameters.AddWithValue("@Items",
xmlItems.ToString());
        xmlParam.SqlDbType = SqlDbType.Xml;

        // execute the command
        connection.Open();
        command.ExecuteNonQuery();
        connection.Close();
      }
  }
}
```

Step 3: Example of using the "ExecuteStoredProcedureWithXML" method. This sample code initializes an array of items, creates a SQL data client connection, and calls the method to pass the array as XML to the SQL stored procedure.

```
class Program
{
  static void Main()
  {
    string[] items = { "Item1", "Item2", "Item3" };

    using (SqlConnection connection = new SqlConnection("sql conn string"))
    {
      DataProcessor processor = new DataProcessor();
      processor.ExecuteStoredProcedureWithXML(connection, items);
    }
  }
}
```

Explanation of the code

XML Creation: Create an "XElement" named "xmlItems" which wraps the individual items inside <Item> tags. This structure matches what the "ProcessItemsFromXML" SQL stored procedure expects.

SQL Command: Instantiate a SQLDataClient "SqlCommand" object and specify the "ProcessItemsFromXML" stored procedure and set its type to "CommandType.StoredProcedure."

Parameter Passing: The XML string is added as a parameter using "AddWithValue" specifying the "SqlDbType.Xml" type because the stored procedure expects XML data as a parameter.

Execution: The command is executed, opening and closing the connection as needed.

Conclusion

Using XML to pass arrays to SQL stored procedures from C# is an effective and flexible approach that allows you to encapsulate multiple values into a single parameter, reducing the overhead of multiple database calls and is useful for batch processing.

http://www.seandrew.info - https://www.linkedin.com/in/seanmdrew