http://www.sqlservercentral.com/articles/XML/61618/
Printed 2007/12/05 10:59AM

XML Workshop XII - Parsing a delimited string

By Jacob Sebastian, 2007/11/29

Introduction

If you have read most of my articles in *XML Workshop* series, you might have figured that I am a strong supporter of *XML*. There had been a lot of debates on *TO-XML* or *NOT-TO-XML* for quite some time. But I guess it is very much subjective. The question should be answered based on a specific task at hand. So the real question is not *TO-XML* or *NOT-TO-XML*. But the question should be '*TO-XML* or *NOT-TO-XML* for the implementation of a specific task'. When I have a specific task at hand, based on various factors I might decide to go for an *XML* approach or an approach without *XML*.

Some times, we take a specific approach even though we know there is a better way of doing the given task. But again, you might have your own reasons for that. So it does not mean that a good technology should always be used. There are so many factors that might influence the Project Manager or the Tech Lead from deciding an approach suitable for the specific task.

We started working with *SQL Server 2005* right from the initial days. I was very much fascinated by the *XML* support. I knew how to use *XQuery* in *TSQL*. We had many stored procedures which accepted string parameters in *XML* format. Those stored procedures used *OPENXML()* to transform the *XML* string to a result set and read values from the result set. I know, I could have used an *XML* variable instead of a string in *XML* structure and used *XQuery* instead of *OPENXML()*. That would have been a better choice given the *XML* support in *SQL Server 2005*. But we still continued with the old approach for a few more months. The reason why we did not use *XQuery* initially was because not all developers were trained enough to work with *XQuery* at that time. We had been very busy and had been totally focussing on getting the modules completed and hence thought of moving ahead with the previous approach and refactoring the code at a later stage.

The reason why I brought you to this discussion is that, I believe it is incorrect to say that 'X is the best approach'. But the correct way is to say 'X is the best approach for Y task given A, B and C conditions'. 'A, B and C' may be the project dead line, size of the application, volume of changes needed, skill level of the programmers and the like.

The goal of my *XML workshop* is to expose *XML* capabilities of *SQL Server 2005* from various angles and give you examples which you might be able to use in your applications right when you decide to go in the *XML* way.

Parsing a delimited string

The most common approach seen in the Internet code samples is to use *PATINDEX* in a *WHILE* loop and use *SUBSTRING()* to chop each piece of data. This data is inserted to a memory table or temp table and consumed. I did a search in Google and found the following articles which explains how to tokenize/parse a delimited string.

- Tokenizing a String Using PARSENAME

- Splitting a string of unlimited length
- Quick T-SQL to parse a delimited string
- T-SQL string parsing

This article presents a different approach which splits a delimited string using the *XML* approach. The following example adds *XML* tags to a comma separated string and retrieves a result set using *XQuery*. [code]

```
/*
Here is the string that we need to split
*/
DECLARE @str VARCHAR(100)
SET @str = '0001,0002,0003,0004,0005'

/*
I am converting the string to an XML structure by
inserting XML tags.
*/
DECLARE @x XML
SET @x = '<i>' + REPLACE(@str, ',', '</i><i>') + '</i>'

/*
Now we can apply XQuery to return a result set
*/
SELECT x.i.value('.', 'VARCHAR(4)') AS Item
    FROM @x.nodes('//i') x(i)

/*
Item
----
0001
0002
0003
0004
0005

(5 row(s) affected)
*/
```

If you want to reuse this code, probably it is a good idea to create a function which does this. Here is an example. [code]

```
CREATE FUNCTION dbo.SplitString
(
    @str VARCHAR(MAX),
    @delimeter CHAR(1)
)
RETURNS @ret TABLE (Token VARCHAR(MAX))
AS
BEGIN

DECLARE @x XML
SET @x = '<t>' + REPLACE(@str, @delimeter, '</t><t>') + '</t>'

INSERT INTO @ret
    SELECT x.i.value('.', 'VARCHAR(MAX)') AS token
    FROM @x.nodes('//t') x(i)

```

```
17 RETURN
18 END
```

It may not be a good approach to use *VARCHAR(MAX)*. If you do not expect really large strings and tokens, it is better to use a fixed size that will suite your specific data requirements.

As a last step, let us test the function we just created. [code]

```
1 SELECT * FROM dbo.SplitString('0001|0002|0003|0004|0005', '|')
2 SELECT * FROM dbo.SplitString('0001,0002,0003,0004,0005', ',')
```

Conclusions

Huh! Yet another way of splitting a string.
