**http://www.sqlservercentral.com/articles/XML/64135/**
Printed 2008/10/01 11:58AM

# XML Workshop XXII - A TSQL RSS Library

## By Jacob Sebastian, 2008/08/20

## Introduction

In the last few sessions of XML Workshop we had been looking at ways of generating *RSS/ATOM* Feeds. You can find the previous sessions here. We have seen how to generate *RSS* and *ATOM* feeds in SQL Server 2005 as well as 2000. In the previous sessions, we have seen how to generate *RSS* and *ATOM* feeds using *FOR XML PATH* as well as *FOR XML EXPLICIT*. If you are working with SQL Server 2005 (and above), you can take advantage of *FOR XML PATH* and if you are still in SQL Server 2000, you can use *FOR XML EXPLICIT*.

Though we have seen two versions of the source code for *RSS* and *ATOM* each, it would be often a difficult task to write the *TSQL* code to generate a correct *RSS* or *ATOM* feed taking data from a given set of tables. To make this task easier, in this session, we will create a function that generates an *RSS* feed from a given channel and item information. We will create a function that accepts two *XML* parameters (channel and items) and generate the required feed structure and returns an *XML* document.

So the focus of this sessions will be writing a function that accepts two XML parameters containing **channel** and **item** information and generates an *RSS* 2.0 feed. We will be able to call the function as in the given example.

```
-- declare the variables
DECLARE @ch XML, @itm XML

-- create an XML document with channel information
SELECT @ch = (
    SELECT * FROM ChannelTable
    FOR XML PATH(''), ROOT('Channel')
)

-- create an XML document with items information
SELECT @itm = (
    SELECT * FROM Products
    FOR XML PATH ('Items'), ROOT('Item')
)

-- generate the feed
SELECT dbo.GenerateRss20( @ch, @itm )
```

## Function that generates RSS feed

Let us look at the function that generates an *RSS* Feed. The code is pretty much the same as what we developed in the previous sessions. The only difference is the part that transforms the *XML* parameters to virtual tables and runs a *FOR XML PATH* query on it to produce the *RSS* Feed. Here is the definition of the function.

```
CREATE FUNCTION GenerateRss20
```

```
(
    @ch XML,     -- Channel Information
    @itm XML     -- Item Information
)
RETURNS XML
AS
BEGIN
    -- This is the variable that will hold the result (RSS feed)
    DECLARE @rss XML

    /*
        To make the process easier, let us transform Channel and Item
        information to a virtual table using CTE.
    */
    ;WITH channel AS (
        SELECT
            c.value('Title[1]','VARCHAR(500)') AS Title,
            c.value('Link[1]','VARCHAR(500)') AS Link,
            c.value('Description[1]','VARCHAR(MAX)') AS Description,
            c.value('Webmaster[1]','VARCHAR(50)') AS Webmaster,
            c.value('Language[1]','VARCHAR(20)') AS Language,
            c.value('ImageUrl[1]','VARCHAR(500)') AS ImageUrl,
            c.value('ImageTitle[1]','VARCHAR(500)') AS ImageTitle,
            c.value('ImageLink[1]','VARCHAR(500)') AS ImageLink,
            c.value('ImageWidth[1]','INT') AS ImageWidth,
            c.value('ImageHeight[1]','INT') AS ImageHeight,
            c.value('CopyRight[1]','VARCHAR(100)') AS CopyRight,
            c.value('LastBuildDate[1]','DATETIME') AS LastBuildDate,
            c.value('Ttl[1]','INT') AS Ttl
        FROM @ch.nodes('/Channel') ch(c)
    ), items AS (
        SELECT
            i.value('Title[1]','VARCHAR(500)') AS Title,
            i.value('Link[1]','VARCHAR(500)') AS Link,
            i.value('Description[1]','VARCHAR(MAX)') AS Description,
            i.value('Guid[1]','VARCHAR(500)') AS Guid,
            i.value('PubDate[1]','DATETIME') AS PubDate
        FROM @itm.nodes('/Item/Items') itm(i)
    )
    /*
        Generate the RSS feed and assign to the local variable
    */
    SELECT @rss = (
        SELECT
            '2.0' AS '@version',
            (
                SELECT
                    Title AS title,
                    Link AS link,
                    Description AS description,
                    Webmaster AS webMaster,
                    ISNULL(Language, 'en-us') AS language,
                    ImageUrl AS 'image/url',
                    ImageTitle AS 'image/title',
                    ImageLink AS 'image/link',
                    ImageWidth AS 'image/width',
                    ImageHeight AS 'image/height',
                    CopyRight AS copyright,
                    LEFT(DATENAME(dw, ISNULL(LastBuildDate,GETDATE())),3) + ', ' +

STUFF(CONVERT(nvarchar,ISNULL(LastBuildDate,GETDATE()),113),21,4,' GMT')
                        AS lastBuildDate,
                    Ttl AS ttl,
                    (
```

```
                        SELECT
                            Title AS title,
                            Link AS link,
                            Description AS description,
                            CASE
                                WHEN ISNULL(guid, Link) IS NULL THEN NULL
                                ELSE 'true'
                            END AS 'guid/@isPermaLink',
                            ISNULL(Guid, Link) AS guid,
                            LEFT(DATENAME(dw, ISNULL(PubDate,GETDATE())),3) + ', ' +

STUFF(CONVERT(nvarchar,ISNULL(PubDate,GETDATE()),113),21,4,' GMT')
                                AS pubDate
                        FROM Items FOR XML PATH('item'), TYPE
                    )
                FROM channel
                FOR XML PATH('channel'), TYPE
            )
        FOR XML PATH('rss')
    )
    -- return the feed
    RETURN @rss
END
```

## Invoking The Function

We have the function ready. Let us see a few examples that invoke the function and generate RSS feeds. Here is a basic example.

```
-- declare variables
DECLARE @ch XML, @itm XML

-- Create an XML document with channel information
SELECT @ch = (
    SELECT
        'TSQL RSS Library' AS Title,
        'http://www.sqlserverandxml.com' AS Link,
        'A TSQL RSS Library to help generating RSS 2.0 feeds' AS Description
    FOR XML PATH(''), ROOT('Channel')
)

-- Create an XML document with item information
SELECT @itm = (
    SELECT
        'Item 1' AS Title,
        'http://www.sqlserverandxml.com/1' AS Link,
        'This is Item 1' AS Description
    FOR XML PATH ('Items'), ROOT('Item')
)

-- generate the feed
SELECT dbo.GenerateRss20( @ch, @itm )
```

```
<rss version="2.0">
  <channel>
    <title>TSQL RSS Library</title>
    <link>http://www.sqlserverandxml.com</link>
    <description>A TSQL RSS Library to help generating RSS 2.0 feeds</description>
    <language>en-us</language>
    <lastBuildDate>Sat, 05 Jul 2008 15:07:45 GMT</lastBuildDate>
    <item>
      <title>Item 1</title>
```

```
      <link>http://www.sqlserverandxml.com/1</link>
      <description>This is Item 1</description>
      <guid isPermaLink="true">http://www.sqlserverandxml.com/1</guid>
      <pubDate>Sat, 05 Jul 2008 15:07:45 GMT</pubDate>
    </item>
  </channel>
</rss>
```

## A Real Life Example

We just saw a basic sample that generates an *RSS* feed using the function we created. Let us now look at a real life example. We will use the **pubs** sample database for this example. Connect to the **pubs** database and create the function.

Assume that we need to generate an *RSS* feed for each author. The feed will contain information about the books written by each author. For the purpose of our example, we will take author **Green Marjorie**.

We need to create two *XML* variables before we can call the function. The first *XML* variable should contain **channel** information and the second should contain **item** information. Let us create the an *XML* document with **channel** information for author **Green Marjorie**. The information of authors is stored in the table **'Authors'**. The *Author ID* of **'Green Marjorie'** is '*213-46-8915*' and we will use it for identifying the correct row from the **'Authors'** table.

```
DECLARE @ch XML

SELECT @ch = (
    SELECT
        au_lname + ' ' + au_fname + '''s Books' AS Title,
        'http://www.sqlserverandxml.com/books/' + au_id AS Link,
        'Books written by ' + au_lname + ' ' + au_fname AS Description
    FROM authors WHERE au_id = '213-46-8915'
    FOR XML PATH(''), ROOT('Channel')
)
```

Now, let us find information about the books of the above author. ID of books written by each author is stored in the table **'TitleAuthor'**. Details of the book is stored in the table **'Titles'**. Let us link these tables and retrieve information about the books written by **'Green Marjorie'**.

```
DECLARE @itm XML

-- Create an XML document with item information
SELECT @itm = (
    SELECT
        t.title AS Title,
        'http://www.sqlserverandxml.com/books/mg/' + t.title_id AS Link,
        t.notes AS Description
    FROM titleauthor ta
    INNER JOIN titles t ON
        ta.title_id = t.title_id
        AND ta.au_id = '213-46-8915'
    FOR XML PATH ('Items'), ROOT('Item')
)
```

The following code generates an RSS feed containing details of books written by **Green Marjorie.**

```
-- declare variables
DECLARE @ch XML, @itm XML
```

```
-- Create an XML document with channel information
SELECT @ch = (
    SELECT
        au_lname + ' ' + au_fname + '''s Books' AS Title,
        'http://www.sqlserverandxml.com/books/' + au_id AS Link,
        'Books written by ' + au_lname + ' ' + au_fname AS Description
    FROM authors WHERE au_id = '213-46-8915'
    FOR XML PATH(''), ROOT('Channel')
)

-- Create an XML document with item information
SELECT @itm = (
    SELECT
        t.title AS Title,
        'http://www.sqlserverandxml.com/books/mg/' + t.title_id AS Link,
        t.notes AS Description
    FROM titleauthor ta
    INNER JOIN titles t ON
        ta.title_id = t.title_id
        AND ta.au_id = '213-46-8915'
    FOR XML PATH ('Items'), ROOT('Item')
)

-- generate the feed
SELECT dbo.GenerateRss20( @ch, @itm )
```

```
<rss version="2.0">
  <channel>
    <title>Green Marjorie's Books</title>
    <link>http://www.sqlserverandxml.com/books/213-46-8915</link>
    <description>Books written by Green Marjorie</description>
    <language>en-us</language>
    <lastBuildDate>Sat, 05 Jul 2008 15:26:48 GMT</lastBuildDate>
    <item>
      <title>The Busy Executive's Database Guide</title>
      <link>http://www.sqlserverandxml.com/books/mg/BU1032</link>
      <description>
        An overview of available database systems with
        emphasis on common business applications. Illustrated.
      </description>
      <guid isPermaLink="true">
        http://www.sqlserverandxml.com/books/mg/BU1032
      </guid>
      <pubDate>Sat, 05 Jul 2008 15:26:48 GMT</pubDate>
    </item>
    <item>
      <title>You Can Combat Computer Stress!</title>
      <link>http://www.sqlserverandxml.com/books/mg/BU2075</link>
      <description>
        The latest medical and psychological techniques for living
        with the electronic office. Easy-to-understand explanations.
      </description>
      <guid isPermaLink="true">
        http://www.sqlserverandxml.com/books/mg/BU2075
      </guid>
      <pubDate>Sat, 05 Jul 2008 15:26:48 GMT</pubDate>
    </item>
  </channel>
</rss>
```

Note that the nodes of the *XML* parameters should follow certain naming rules. The function identifies the elements by applying an exact match on the name and hence the elements of your *XML* parameter should be

correctly named. The **channel** parameter recognizes the following elements.

- `Title`
- `Link`
- `Description`
- `Webmaster`
- `Language`
- `ImageUrl`
- `ImageTitle`
- `ImageLink`
- `ImageWidth`
- `ImageHeight`
- `CopyRight`
- `LastBuildDate`
- `Ttl`

The element names should *EXACTLY* match with the list given above. The function will ignore any elements that it cannot recognize. If you mis-spell a few elements, a feed will still be generated, but it may not be a valid *RSS* feed (as it would be missing some of the elements).

Just like the **'channel'** parameter, the **'item'** parameter too, expects the elements to follow certain naming rules. The **'item'** parameter expects the following elements

- `Title`
- `Link`
- `Description`
- `Guid`
- `PubDate`

## Conclusions

In this session, we created a function that generates an *RSS* 2.0 feed. The function takes two *XML* parameters containing the channel and item information. A feed is then generated based on the information stored in the *XML* parameters.

---