http://www.sqlservercentral.com/articles/XML/62931/
Printed 2008/06/05 06:31PM

XML Workshop XIX - Generating an ATOM 1.0 Feed

By Jacob Sebastian, 2008/04/25

Introduction

In the previous article, I had presented an example that generates an RSS 2.0 feed using TSQL. In this session, we will generate an ATOM 1.0 feed. ATOM is another popular feed format and you can find the specification here.

Most of the times, web applications generate feeds at the application layer. The feed generator would execute a query/stored-procedure, fetch the data and generate the XML document using custom application code or a third party library like ATOM.NET. The purpose of this session is to look deeper into the XML capabilities of SQL Server 2005 and see if it could generate an ATOM 1.0 with a FOR XML operation. The examples and code given in this article are purely for the purpose of explaining how XML data can be generated in TSQL.

In the previous article, I had explained how to validate a web feed with an online feed validator. Though there are several online feed validators available, we will use FeedValidator.org to validate the feeds that we generate in this session. You could try the feeds with other feed validators and get back to me if you find an issue.

Sample Feed

Here is the output that we expect to generate by the end of this LAB.

```xml
<feed xmlns="http://www.w3.org/2005/Atom">
  <title type="html">Welcome to XML Workshop</title>
  <subtitle type="html">
    A collection of short articles on
    SQL Server and XML
  </subtitle>
  <id>
    http://www.sqlserverandxml.com-a.googlepages.com/TSQLAtom10.xml
  </id>
  <link xmlns="http://www.w3.org/2005/Atom" rel="alternate"
        type="text/html" href="http://www.sqlserverandxml.com/" />
  <link xmlns="http://www.w3.org/2005/Atom" rel="self"
        type="application/atom+xml"
        href="http://www.sqlserverandxml.com-a.googlepages.com/TSQLAtom10.xml" />
  <generator uri="http://www.sqlserverandxml.com/" version="1.0">
    FOR XML
  </generator>
  <updated>2005-10-14T03:17:00Z</updated>
  <entry xmlns="http://www.w3.org/2005/Atom">
    <title>Sales Order Workshop</title>
    <link rel="alternate" type="text/html"
          href="http://www.sqlserverandxml.com-a.googlepages.com
/salesorderworkshop">
      http://www.sqlserverandxml.com-a.googlepages.com/salesorderworkshop
    </link>
    <id>http://www.sqlserverandxml.com-a.googlepages.com/salesorderworkshop</id>
    <published>2005-11-24T00:25:00Z</published>
    <updated>2005-11-24T00:25:00Z</updated>
    <content>
      A series of 4 articles that explain
      how to pass variable number of parameters to a stored procedure
      using XML
    </content>
    <author>
      <name>Jacob Sebastian</name>
      <uri>http://www.sqlserverandxml.com</uri>
    </author>
  </entry>
  <entry xmlns="http://www.w3.org/2005/Atom">
    <title>FOR XML Workshop</title>
    <link rel="alternate" type="text/html"
          href="http://www.sqlserverandxml.com-a.googlepages.com/forxmlworkshop">
      http://www.sqlserverandxml.com-a.googlepages.com/forxmlworkshop
    </link>
    <id>http://www.sqlserverandxml.com-a.googlepages.com/forxmlworkshop</id>
    <published>2005-10-14T02:17:00Z</published>
    <updated>2005-10-14T02:17:00Z</updated>
    <content>
      A collection of short articles that
      explain how to generate XML output using TSQL keyword FOR
      XML.
    </content>
    <author>
      <name>Jacob Sebastian</name>
      <uri>http://www.sqlserverandxml.com/</uri>
    </author>
  </entry>
</feed>
```

Before we proceed further, we need to make sure that the XML output that we intend to generate is a valid ATOM 1.0 document. You could test this by using feedvalidator.org. Open a browser window and navigate to feedvalidator.org. Enter the url of the above sample ATOM 1.0 feed and click on the "validate" button.

Sample Tables and Data

Let us create two tables to store the data needed for this LAB. We need one table to store the information about the *feed* element and another table for storing the data of each *entry* in the feed. Here is the script of those tables.

```
-- table for the feed information
CREATE TABLE feed(
    title VARCHAR(100),
    subtitle VARCHAR(200),
    id VARCHAR(100),
    link VARCHAR(100),
    generator VARCHAR(20),
    updated    DATETIME )
GO

-- table to store the entries
CREATE TABLE entry(
    title VARCHAR(100),
    link VARCHAR(100),
    published DATETIME,
    updated DATETIME,
    content VARCHAR(1000),
    authorname VARCHAR(30),
    authorurl VARCHAR(100))
GO
```

Here is the code to populate the tables with some sample data

```
-- populate the 'feed' table
```

```
INSERT INTO feed (
    title,
    subtitle,
    id,
    link,
    generator,
    updated )
SELECT
    'Welcome to XML Workshop',
    'A collection of short articles on SQL Server and XML',
    'http://www.sqlserverandxml.com-a.googlepages.com/TSQLAtom10.xml',
    'http://www.sqlserverandxml.com/',
    'FOR XML',
    '2005-10-14T03:17:00'

-- populate the 'entry' table
INSERT INTO entry(
    title,
    link,
    published,
    updated,
    content,
    authorname,
    authorurl )
SELECT
    'Sales Order Workshop',
    'http://www.sqlserverandxml.com-a.googlepages.com/salesorderworkshop',
    '2005-11-24T00:25:00',
    '2005-11-24T00:25:00',
    'A series of 4 articles which explain
      how to pass variable number of parameters to a stored procedure
      using XML',
    'Jacob Sebastian',
    'http://www.sqlserverandxml.com'
UNION ALL
SELECT
    'FOR XML Workshop',
    'http://www.sqlserverandxml.com-a.googlepages.com/forxmlworkshop',
    '2005-10-14T02:17:00',
    '2005-10-14T02:17:00',
    'A collection of short articles that
      explain how to generate XML output using TSQL keyword FOR
      XML.',
    'Jacob Sebastian',
    'http://www.sqlserverandxml.com/'
```

Generating the feed

Let us start with the entry element, which is pretty much easy. Here is the code that generates the entry elements.

```
SELECT
    title,
    'alternate' AS 'link/@rel',
    'text/html' AS 'link/@type',
    link AS 'link/@href',
    link,
    link AS 'id',
    published,
    updated,
    content,
    authorname AS 'author/name',
    authorurl AS 'author/uri'
FROM entry FOR XML PATH('entry'), TYPE
```

The above code produces the following XML document.

```
<entry>
  <title>Sales Order Workshop</title>
  <link rel="alternate" type="text/html"
        href="http://www.sqlserverandxml.com-a.googlepages.com/salesorderworkshop">
    http://www.sqlserverandxml.com-a.googlepages.com/salesorderworkshop
  </link>
  <id>
    http://www.sqlserverandxml.com-a.googlepages.com/salesorderworkshop
  </id>
  <published>2005-11-24T00:25:00</published>
  <updated>2005-11-24T00:25:00</updated>
  <content>
    A series of 4 articles which explain
    how to pass variable number of parameters to a stored procedure
    using XML
  </content>
  <author>
    <name>Jacob Sebastian</name>
    <uri>http://www.sqlserverandxml.com</uri>
  </author>
</entry>
<entry>
  <title>FOR XML Workshop</title>
  <link rel="alternate" type="text/html"
        href="http://www.sqlserverandxml.com-a.googlepages.com/forxmlworkshop">
    http://www.sqlserverandxml.com-a.googlepages.com/forxmlworkshop
  </link>
  <id>http://www.sqlserverandxml.com-a.googlepages.com/forxmlworkshop</id>
  <published>2005-10-14T02:17:00</published>
  <updated>2005-10-14T02:17:00</updated>
  <content>
    A collection of short articles that
    explain how to generate XML output using TSQL keyword FOR
    XML.
  </content>
  <author>
    <name>Jacob Sebastian</name>
    <uri>http://www.sqlserverandxml.com/</uri>
  </author>
</entry>
```

Though the XML looks good, there is a problem. The format of the date values (published and updated) are not correct. ATOM 1.0 requires that the date value should be in RFC 3339 date format. So we need to format the date value to a valid RFC 3339 date value. Here is the modified version of the code.

```
SELECT
    title,
    'alternate' AS 'link/@rel',
    'text/html' AS 'link/@type',
    link AS 'link/@href',
    link,
    link AS 'id',
    CONVERT(nvarchar,published,127) + 'Z' AS published,
    CONVERT(nvarchar,updated,127) + 'Z' AS updated,
    content,
    authorname AS 'author/name',
    authorurl AS 'author/uri'
FROM entry FOR XML PATH('entry'), TYPE
```

Here is the corrected XML.

```
<entry>
  <title>Sales Order Workshop</title>
  <link rel="alternate" type="text/html"
        href="http://www.sqlserverandxml.com-a.googlepages.com/salesorderworkshop">
    http://www.sqlserverandxml.com-a.googlepages.com/salesorderworkshop
  </link>
  <id>http://www.sqlserverandxml.com-a.googlepages.com/salesorderworkshop</id>
  <published>2005-11-24T00:25:00Z</published>
  <updated>2005-11-24T00:25:00Z</updated>
  <content>
    A series of 4 articles which explain
    how to pass variable number of parameters to a stored procedure
    using XML
  </content>
  <author>
    <name>Jacob Sebastian</name>
    <uri>http://www.sqlserverandxml.com</uri>
  </author>
</entry>
<entry>
  <title>FOR XML Workshop</title>
  <link rel="alternate" type="text/html"
        href="http://www.sqlserverandxml.com-a.googlepages.com/forxmlworkshop">
    http://www.sqlserverandxml.com-a.googlepages.com/forxmlworkshop
  </link>
  <id>http://www.sqlserverandxml.com-a.googlepages.com/forxmlworkshop</id>
  <published>2005-10-14T02:17:00Z</published>
  <updated>2005-10-14T02:17:00Z</updated>
  <content>
    A collection of short articles that
    explain how to generate XML output using TSQL keyword FOR
    XML.
  </content>
  <author>
    <name>Jacob Sebastian</name>
    <uri>http://www.sqlserverandxml.com/</uri>
  </author>
</entry>
```

Now, let us write the query to generate the "feed" element.

```
SELECT
    'html' AS 'title/@type',
    title,
    'html' AS 'subtitle/@type',
    subtitle,
    id,
    (
        SELECT
            'alternate' AS 'link/@rel',
            'text/html' AS 'link/@type',
            link AS 'link/@href'
        FROM feed FOR XML PATH(''), TYPE
    ),
    (
        SELECT
            'self' AS 'link/@rel',
            'application/atom+xml' AS 'link/@type',
            id AS 'link/@href'
        FROM feed FOR XML PATH(''), TYPE
    ),
    link AS 'generator/@uri',
    '1.0' AS 'generator/@version',
    generator,
    CONVERT(VARCHAR(20),updated,127) + 'Z' AS updated
FROM feed
FOR XML PATH('feed'),TYPE
```

```
<feed>
  <title type="html">Welcome to XML Workshop</title>
  <subtitle type="html">
    A collection of short articles on SQL Server and XML
  </subtitle>
  <id>
    http://www.sqlserverandxml.com-a.googlepages.com/TSQLAtom10.xml
  </id>
  <link rel="alternate" type="text/html"
        href="http://www.sqlserverandxml.com/" />
  <link rel="self" type="application/atom+xml"
        href="http://www.sqlserverandxml.com-a.googlepages.com/TSQLAtom10.xml" />
  <generator uri="http://www.sqlserverandxml.com/" version="1.0">
    FOR XML
  </generator>
  <updated>2005-10-14T03:17:00Z</updated>
</feed>
```

Unlike RSS 2.0 feeds, ATOM 1.0 should have a namespace declaration. We could add a namespace declaration by adding *WITH XMLNAMESPACES* clause. Here is how we will achieve this.

```
;WITH XMLNAMESPACES(
    DEFAULT 'http://www.w3.org/2005/Atom'
)
SELECT
    'html' AS 'title/@type',
    title,
    'html' AS 'subtitle/@type',
    subtitle,
    id,
    (
        SELECT
            'alternate' AS 'link/@rel',
            'text/html' AS 'link/@type',
            link AS 'link/@href'
        FROM feed FOR XML PATH(''), TYPE
    ),
    (
        SELECT
            'self' AS 'link/@rel',
            'application/atom+xml' AS 'link/@type',
            id AS 'link/@href'
        FROM feed FOR XML PATH(''), TYPE
    ),
    link AS 'generator/@uri',
    '1.0' AS 'generator/@version',
    generator,
    CONVERT(VARCHAR(20),updated,127) + 'Z' AS updated
FROM feed
FOR XML PATH('feed'),TYPE
```

```
<feed xmlns="http://www.w3.org/2005/Atom">
  <title type="html">Welcome to XML Workshop</title>
  <subtitle type="html">
    A collection of short articles on SQL Server and XML
  </subtitle>
  <id>
    http://www.sqlserverandxml.com-a.googlepages.com/TSQLAtom10.xml
  </id>
  <link xmlns="http://www.w3.org/2005/Atom"
        rel="alternate" type="text/html"
        href="http://www.sqlserverandxml.com/" />
  <link xmlns="http://www.w3.org/2005/Atom"
        rel="self" type="application/atom+xml"
        href="http://www.sqlserverandxml.com-a.googlepages.com/TSQLAtom10.xml" />
  <generator uri="http://www.sqlserverandxml.com/" version="1.0">
    FOR XML
  </generator>
  <updated>2005-10-14T03:17:00Z</updated>
</feed>
```

We have seen how to generate the *feed* element as well as *entry* elements. Now it is time to merge the queries and produce the final result.

```
;WITH XMLNAMESPACES(
    DEFAULT 'http://www.w3.org/2005/Atom'
)
SELECT
    'html' AS 'title/@type',
    title,
    'html' AS 'subtitle/@type',
    subtitle,
    id,
    (
        SELECT
            'alternate' AS 'link/@rel',
            'text/html' AS 'link/@type',
            link AS 'link/@href'
        FROM feed FOR XML PATH(''), TYPE
    ),
    (
        SELECT
            'self' AS 'link/@rel',
            'application/atom+xml' AS 'link/@type',
            id AS 'link/@href'
        FROM feed FOR XML PATH(''), TYPE
    ),
    link AS 'generator/@uri',
    '1.0' AS 'generator/@version',
    generator,
    CONVERT(VARCHAR(20),updated,127) + 'Z' AS updated,
    (
        SELECT
            title,
            'alternate' AS 'link/@rel',
            'text/html' AS 'link/@type',
            link AS 'link/@href',
            link,
            link AS 'id',
            CONVERT(nvarchar,published,127) + 'Z' AS published,
            CONVERT(nvarchar,updated,127) + 'Z' AS updated,
            content,
            authorname AS 'author/name',
            authorurl AS 'author/uri'
        FROM entry FOR XML PATH('entry'), TYPE
    )
    FROM feed
FOR XML PATH('feed'),TYPE
```

```
<feed xmlns="http://www.w3.org/2005/Atom">
  <title type="html">Welcome to XML Workshop</title>
  <subtitle type="html">
    A collection of short articles on SQL Server and XML
  </subtitle>
  <id>
    http://www.sqlserverandxml.com-a.googlepages.com/TSQLAtom10.xml
  </id>
  <link xmlns="http://www.w3.org/2005/Atom"
        rel="alternate" type="text/html"
        href="http://www.sqlserverandxml.com/" />
  <link xmlns="http://www.w3.org/2005/Atom"
        rel="self" type="application/atom+xml"
        href="http://www.sqlserverandxml.com-a.googlepages.com/TSQLAtom10.xml" />
  <generator uri="http://www.sqlserverandxml.com/" version="1.0">
    FOR XML
  </generator>
  <updated>2005-10-14T03:17:00Z</updated>
  <entry xmlns="http://www.w3.org/2005/Atom">
    <title>Sales Order Workshop</title>
    <link rel="alternate" type="text/html"
```

```
          href="http://www.sqlserverandxml.com-a.googlepages.com
/salesorderworkshop">
      http://www.sqlserverandxml.com-a.googlepages.com/salesorderworkshop
    </link>
    <id>http://www.sqlserverandxml.com-a.googlepages.com/salesorderworkshop</id>
    <published>2005-11-24T00:25:00Z</published>
    <updated>2005-11-24T00:25:00Z</updated>
    <content>
      A series of 4 articles which explain
      how to pass variable number of parameters to a stored procedure
      using XML
    </content>
    <author>
      <name>Jacob Sebastian</name>
      <uri>http://www.sqlserverandxml.com</uri>
    </author>
  </entry>
  <entry xmlns="http://www.w3.org/2005/Atom">
    <title>FOR XML Workshop</title>
    <link rel="alternate" type="text/html"
          href="http://www.sqlserverandxml.com-a.googlepages.com/forxmlworkshop">
      http://www.sqlserverandxml.com-a.googlepages.com/forxmlworkshop
    </link>
    <id>http://www.sqlserverandxml.com-a.googlepages.com/forxmlworkshop</id>
    <published>2005-10-14T02:17:00Z</published>
    <updated>2005-10-14T02:17:00Z</updated>
    <content>
      A collection of short articles that
      explain how to generate XML output using TSQL keyword FOR
      XML.
    </content>
    <author>
      <name>Jacob Sebastian</name>
      <uri>http://www.sqlserverandxml.com/</uri>
    </author>
  </entry>
</feed>
```

We have got a valid ATOM 1.0 feed at this point. This feed validates successfully with a few feed validators that I tried. However, you might notice something strange with this feed. Usually you will see ATOM 1.0 feeds having namespace declaration only in the *feed* element. In our case, we have namespace declaration added to the *link* and *entry* elements too. This is the way 'WITH XMLNAMESPACES' works. There is no direct way to get rid of this.

A feed validator does not seem to complain about this. But I am not sure if all feed readers will accept this. Technically this should not be a problem. If you want to get rid of this additional namespace declarations on the child elements, you might need to do some string manipulation like in the example given below.

```
SELECT CAST( '<feed xmlns="http://www.w3.org/2005/Atom">' +
CAST(
    (SELECT
        'html' AS 'title/@type',
        title,
        'html' AS 'subtitle/@type',
        subtitle,
        id,
        (
            SELECT
                'alternate' AS 'link/@rel',
                'text/html' AS 'link/@type',
                link AS 'link/@href'
            FROM feed FOR XML PATH(''), TYPE
        ),
        (
            SELECT
```

```
                'self' AS 'link/@rel',
                'application/atom+xml' AS 'link/@type',
                id AS 'link/@href'
            FROM feed FOR XML PATH(''), TYPE
        ),
        link AS 'generator/@uri',
        '1.0' AS 'generator/@version',
        generator,
        CONVERT(VARCHAR(20),updated,127) + 'Z' AS updated,
        (
            SELECT
                title,
                'alternate' AS 'link/@rel',
                'text/html' AS 'link/@type',
                link AS 'link/@href',
                link,
                link AS 'id',
                CONVERT(nvarchar,published,127) + 'Z' AS published,
                CONVERT(nvarchar,updated,127) + 'Z' AS updated,
                content,
                authorname AS 'author/name',
                authorurl AS 'author/uri'
            FROM entry FOR XML PATH('entry'), TYPE
        )
        FROM feed
    FOR XML PATH(''),TYPE)
AS NVARCHAR(MAX)) + '</feed>' AS XML)
```

```
<feed xmlns="http://www.w3.org/2005/Atom">
  <title type="html">Welcome to XML Workshop</title>
  <subtitle type="html">
    A collection of short articles on SQL Server and XML
  </subtitle>
  <id>http://www.sqlserverandxml.com-a.googlepages.com/TSQLAtom10.xml</id>
  <link rel="alternate" type="text/html"
        href="http://www.sqlserverandxml.com/" />
  <link rel="self" type="application/atom+xml"
        href="http://www.sqlserverandxml.com-a.googlepages.com/TSQLAtom10.xml" />
  <generator uri="http://www.sqlserverandxml.com/" version="1.0">
    FOR XML
  </generator>
  <updated>2005-10-14T03:17:00Z</updated>
  <entry>
    <title>Sales Order Workshop</title>
    <link rel="alternate" type="text/html"
          href="http://www.sqlserverandxml.com-a.googlepages.com
/salesorderworkshop">
      http://www.sqlserverandxml.com-a.googlepages.com/salesorderworkshop
    </link>
    <id>http://www.sqlserverandxml.com-a.googlepages.com/salesorderworkshop</id>
    <published>2005-11-24T00:25:00Z</published>
    <updated>2005-11-24T00:25:00Z</updated>
    <content>
      A series of 4 articles which explain
      how to pass variable number of parameters to a stored procedure
      using XML
    </content>
    <author>
      <name>Jacob Sebastian</name>
      <uri>http://www.sqlserverandxml.com</uri>
    </author>
  </entry>
  <entry>
    <title>FOR XML Workshop</title>
    <link rel="alternate" type="text/html"
          href="http://www.sqlserverandxml.com-a.googlepages.com/forxmlworkshop">
      http://www.sqlserverandxml.com-a.googlepages.com/forxmlworkshop
    </link>
    <id>http://www.sqlserverandxml.com-a.googlepages.com/forxmlworkshop</id>
    <published>2005-10-14T02:17:00Z</published>
    <updated>2005-10-14T02:17:00Z</updated>
    <content>
      A collection of short articles that
      explain how to generate XML output using TSQL keyword FOR
      XML.
    </content>
    <author>
      <name>Jacob Sebastian</name>
      <uri>http://www.sqlserverandxml.com/</uri>
    </author>
  </entry>
</feed>
```

You could validate this version of the output with feedvalidator.org to make sure that we have a valid ATOM 1.0 feed.

Conclusions

This is yet another session that demonstrates an XML shaping example. We have seen different XML shaping requirements and their implementation in the previous sessions of the XML Workshop. This session explains the basics of generating an ATOM 1.0 feed using TSQL keyword FOR XML PATH. In the next session, I will present some code that explains how to achieve this in SQL Server 2000.