http://www.sqlservercentral.com/articles/XML/61334/
Printed 2007/11/21 06:24AM

XML Workshop XI - Default Namespaces

By Jacob Sebastian, 2007/10/18

Introduction

In the previous workshop (XML Workshop X) we have seen a basic introduction to *XML* Namespaces. We have seen examples which clearly explained why we need to have namespaces in *XML*. In this session we will examine *XML Default Namespaces* and see how to generate *XML* which contains default namespace definitions.

Default Namespace

Default namespace applies to all the un prefixed elements of an *XML* document. A default namespace declaration is usually placed at the root of an *XML* document. In this case, the default namespace is said to have *global scope* and all the un prefixed elements of the *XML* document will belong to the default namespace. It is also quite possible to have a default namespace defined for a specific *XML* element. In this case, the scope of the namespace declaration will last up to the closing tag of the element. Let us look at a few examples.

Here is the XML that we generated in the previous lab. [xml]

```
<Configuration
  xmlns:db="urn:www.dotnetquest.com/DbConnection"
  xmlns:net="urn:www.dotnetquest.com/NetConnection">
  <net:Connection>
    <net:Provider>World Wide Internet Providers</net:Provider>
    <net:Speed>512 KBPS</net:Speed>
  </net:Connection>
  <db:Connection>
    <db:Provider>SQL Client Provider</db:Provider>
    <db:Protocol>TCP/IP</db:Protocol>
    <db:Authentication>Windows</db:Authentication>
  </db:Connection>
</Configuration>
```

Given below is the equivalent version of the same XML which uses default a default namespace. [xml]

```
<!--
Note that the first namespace is declared as default
namespace. Note that no prefix assigned to this namespace
-->
<Configuration
  xmlns="urn:www.dotnetquest.com/DbConnection"
  xmlns:net="urn:www.dotnetquest.com/NetConnection">
  <net:Connection>
    <net:Provider>World Wide Internet Providers</net:Provider>
    <net:Speed>512 KBPS</net:Speed>
  </net:Connection>
```

```
  <!--
  Note that we are not using any prefix here. This element
  and all the child elements will belong to the default
  namespace.-->
  <Connection>
    <Provider>SQL Client Provider</Provider>
    <Protocol>TCP/IP</Protocol>
    <Authentication>Windows</Authentication>
  </Connection>
</Configuration>
```

The above *XML* is equivalent to the previous one that we have seen. The only difference is that the new version of the *XML* makes use of the default namespace declaration. Let us modify the *TSQL* Query that we created in the previous lab, so that it will generate the *XML* structure with a default namespace declaration. If you have not done the previous lab, you need to create the sample tables and populate them. You can find the script here. The following query will generate the *XML* structure that we just discussed. [code]

```
WITH XMLNAMESPACES
(
    -- This is the default namespace
    DEFAULT 'urn:www.dotnetquest.com/DbConnection',
    'urn:www.dotnetquest.com/NetConnection' AS net
)
SELECT
    net.Provider AS 'net:Connection/net:Provider',
    net.Speed AS 'net:Connection/net:Speed',
    -- we don't need the prefix any more
    db.Provider AS 'Connection/Provider',
    db.Protocol AS 'Connection/Protocol',
    db.[Authentication] AS 'Connection/Authentication'
FROM NetConnection net
CROSS JOIN DbConnection db
FOR XML PATH('Configuration')
```

Reading values

Now let us see how to read values from an XML variable which contains namespace information. In the previous labs we have seen several examples of reading values from *XML* variables and columns. We have not seen any example with namespace information so far. Here is the query that reads values from an *XML* variable which contains namespace information. [code]

```
declare @x xml
set @x = '
<Configuration
  xmlns:db="urn:www.dotnetquest.com/DbConnection"
  xmlns:net="urn:www.dotnetquest.com/NetConnection">
  <net:Connection>
    <net:Provider>World Wide Internet Providers</net:Provider>
    <net:Speed>512 KBPS</net:Speed>
  </net:Connection>
  <db:Connection>
    <db:Provider>SQL Client Provider</db:Provider>
    <db:Protocol>TCP/IP</db:Protocol>
    <db:Authentication>Windows</db:Authentication>
  </db:Connection>
</Configuration>
```

```
'
-- read values from the XML variable
SELECT
    x.c.value(
        'declare namespace net="urn:www.dotnetquest.com/NetConnection";
        (net:Connection/net:Provider)[1]', 'varchar(max)')
        AS NetProvider,
    x.c.value(
        'declare namespace net="urn:www.dotnetquest.com/NetConnection";
        (net:Connection/net:Speed)[1]', 'varchar(max)')
        AS Speed,
    x.c.value(
        'declare namespace db="urn:www.dotnetquest.com/DbConnection";
        (db:Connection/db:Provider)[1]', 'varchar(max)')
        AS DbProvider,
    x.c.value(
        'declare namespace db="urn:www.dotnetquest.com/DbConnection";
        (db:Connection/db:Protocol)[1]', 'varchar(max)')
        AS Protocol,
    x.c.value(
        'declare namespace db="urn:www.dotnetquest.com/DbConnection";
        (db:Connection/db:Authentication)[1]', 'varchar(max)')
        AS [Authentication]
    FROM @x.nodes('/Configuration') x(c)
```

Using *WITH XMLNAMESPACES* you can make this query simpler. Here is a different syntax which produces the same results, but using *WITH XMLNAMESPACES*. [code]

```
declare @x xml
set @x = '
<Configuration
  xmlns:db="urn:www.dotnetquest.com/DbConnection"
  xmlns:net="urn:www.dotnetquest.com/NetConnection">
  <net:Connection>
    <net:Provider>World Wide Internet Providers</net:Provider>
    <net:Speed>512 KBPS</net:Speed>
  </net:Connection>
  <db:Connection>
    <db:Provider>SQL Client Provider</db:Provider>
    <db:Protocol>TCP/IP</db:Protocol>
    <db:Authentication>Windows</db:Authentication>
  </db:Connection>
</Configuration>
'
-- read values from the XML variable
;WITH XMLNAMESPACES
(
    'urn:www.dotnetquest.com/NetConnection' AS net,
    'urn:www.dotnetquest.com/DbConnection' AS db
)
SELECT
    x.c.value(
        '(net:Connection/net:Provider)[1]', 'varchar(20)')
        AS NetProvider,
    x.c.value(
        '(net:Connection/net:Speed)[1]', 'varchar(10)')
        AS Speed,
    x.c.value(
        '(db:Connection/db:Provider)[1]', 'varchar(20)')
```

```
    32          AS DbProvider,
    33      x.c.value(
    34          '(db:Connection/db:Protocol)[1]', 'varchar(10)')
    35          AS Protocol,
    36      x.c.value(
    37          '(db:Connection/db:Authentication)[1]', 'varchar(10)')
    38          AS [Authentication]
    39      FROM @x.nodes('/Configuration') x(c)
    40
    41  /*
    42  OUTPUT:
    43
    44  NetProvider          Speed     DbProvider            Protocol    Authentication
    45  -------------------- ---------- -------------------- ---------- -------------
-
    46  World Wide Internet 512 KBPS   SQL Client Provider TCP/IP     Windows
    47
    48  (1 row(s) affected)
    49  */
```

Conclusions

This workshop focussed on explaining *XML NAMESPACES*. We have seen how go generate *XML* which contains namespace information. We then saw how to read values from an XML variable which contains namespace information.
