I've Got the XML â€“ Now What?

OK, I've got the XML Now what can I do with it?!!

In this article, I'm going to show you how can start by extracting XML from your database and finish up with an html file which documents your database tables.Along the way we'll also be touching upon System Views, XSL, XPATH and SSIS.

The goal is to give you a taste of what you can do with your XML, and hopefully inspire you to put XML to use, and pick up where I've left off. The stages are listed below

1. Write the SQL to get the XML.
2. Write the XSL to transform the XML to HTML.
3. Write the SSIS package to make the transformation happen.

To keep things as simple as possible, the example we'll be working through will just create a list of table and field names, but I'll also include source code to extract information about Primary keys, indexes, foreign keys etc..

1. Writing the SQL to get the XML

Figure 1 shows the SQL to extract the table and field information. Figure 2 shows example XML that this generates. The query that extracts the field names is a subquery of the query which extracts the table names. This means the fields appear as child nodes of the parent table.

```
select xTable.name as table_name
 , (
 select col.name from sys.columns col
 where col.object_id=xTable.object_id
 ORDER BY col.column_id
 for XML auto, type
   )
from sys.tables xTable for
XML auto, type
```

Figure 1

```
<ROOT>
 <xTable table_name="Customer">
 <col name="CustomerID" />
 <col name="SalesPersonID" />
 <col name="TerritoryID" />
 <col name="AccountNumber" />
 <col name="CustomerType" />
 <col name="ModifiedDate" />
 <col name="rowguid" />
```

```
 </xTable>
 <xTable table_name="CustomerAddress">
 <col name="CustomerID" />
 <col name="AddressID" />
 <col name="AddressTypeID" />
 <col name="rowguid" />
 </xTable>
</ROOT>
```

Figure 2

2. XSL which transforms the XML into HTML.

The XSL is shown in figure 3. Basically we start with the ROOT node, and navigate down through the document. Inside the root node we put the HTML that we only want to display once in the document -- as we know we will only have one ROOT node in the XML document. The <XSL:apply-templates select="xTable"> will write the text inside the <XSL:template match="xTable"> node, for each xTable node in the XML document. The xTable template in turn calls the col template for each child col node.

You could parallel this to pseudo code like

```
write header

Foreach xTable in ROOT
write something
 Foreach col in xTable
 write something
 Next
Next
write footer
```

I'd recommend you get an XML editor such as XML SPY to help you author XSL sheets. (There is a free home edition, with sufficient functionality.) This will give you access to intellisense, will syntax check your documents, and will execute transformations.

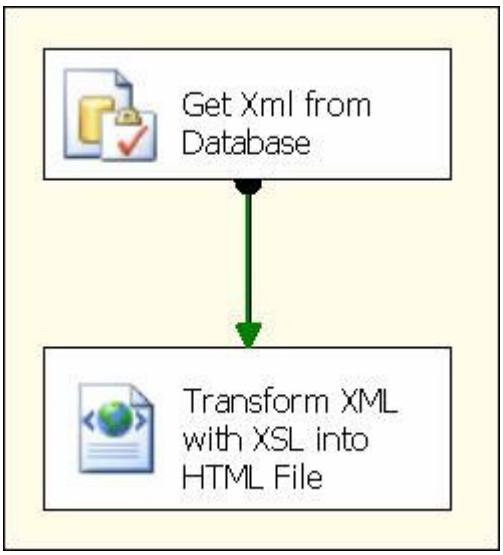
```
 <version="1.0" encoding="UTF-8"?>
<XSL:stylesheet version="2.0" XMLns:XSL="http://www.w3.org/1999/XSL/Transform">
<XSL:template match="/">
 <XSL:apply-templates select="ROOT">
 </XSL:apply-templates>
     </XSL:template>
 <XSL:template match="ROOT">
 <HTML>
    <head>
 <title>Table Generation</title>
</head>
<body>
 <table>
 <XSL:apply-templates select="xTable">
 </XSL:apply-templates>
 </table>
</body>
</HTML>
 </XSL:template>
 <XSL:template match="xTable">
    <tr>
```

```
 <th>
 <XSL:value-of select="@table_name"/>
 </th>
 </tr>
<XSL:apply-templates select="col">
 </XSL:apply-templates>
 </XSL:template>

<XSL:template match="col">
 <tr>
 <td>
 <XSL:value-of select="@name"/>
  </td>
 </tr>
</XSL:template>

</XSL:stylesheet>
```

Figure 3

3. Write the SSIS package to make it happen.

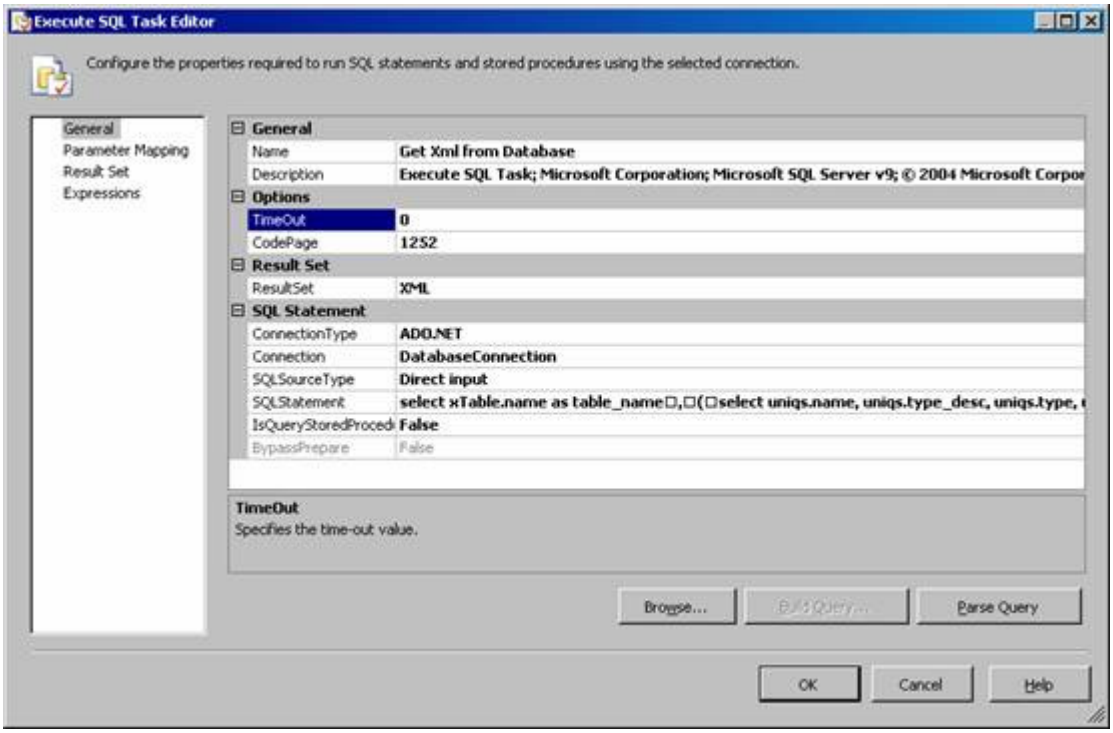
I mentioned earlier that a XML authoring tool can be used to execute a transformation. Sometimes, though, you need a more automated solution. For example, if you want to carry out multiple transformations on the same document or on multiple documents. (e.g. if you wanted to create a separate HTML file for each table.) There are several ways to achieve this. If you don't want to write code, then perhaps SSIS is the easiest method available.



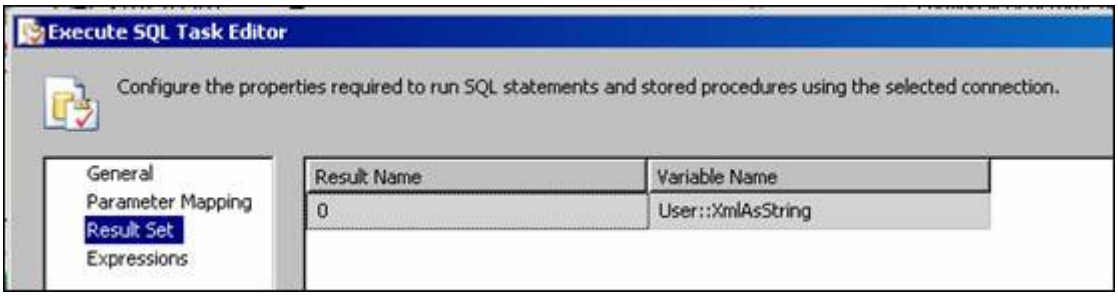
Picture 1 - The SSIS package

1. Execute SQL Task

You need an Execute SQL task with a connection to your database. Make sure the ResultSet is set to XML, SQLSourceType to DirectInput, and paste the SQL statement we constructed earlier into the SQLStatement.



We need to catch the output in a global variable, so create a string variable called XMLAsString, and set the Result Set Variable Name to this. Note that Result Name must be set to zero. (Don't ask me why. It just does!)



2. XML Task

You need an XML Task, with OperationType XSLT (to denote that you're doing a transformation). You need two file connections, one for the XSL file (which you save to the file system) and another for the output HTML file.

Running this package generates an HTML file like the one below.

I agree, it's hardly spectacular. I've included a more sophisticated SQL query and XSL file which will give you the following style of output.

CurrencyRate	
CurrencyRateID	(PK, int, NULL)
CurrencyRateDate	(datetime, NULL)
FromCurrencyCode	(nchar(6), NULL)
ToCurrencyCode	(nchar(6), NULL)
AverageRate	(float, NULL)
EndOfDayRate	(float, NULL)
ModifiedDate	(datetime, NULL)
rowguid	(uniqueidentifier, NULL)
Customer	
CustomerID	(PK, int, NULL)
SalesPersonID	(int, NULL)
TerritoryID	(tinyint, NULL)
AccountNumber	(int, NULL)
CustomerType	(nchar(2), NULL)
ModifiedDate	(datetime, NULL)
rowguid	(uniqueidentifier, NULL)
CustomerAddress	
CustomerID	(PK, int, NULL)
AddressID	(PK, int, NULL)
AddressTypeID	(tinyint, NULL)
rowguid	(uniqueidentifier, NULL)

This is still nothing spectacular, but if you can follow the code and are good at HTML, you can quite easily expand on it to generate much more comprehensive output. I should mention that you don't have to generate HTML. It could be a csv file, or a generate table script. There is an infinite number of possible applications.

If you're interested, you can download some XSL code for more sophisticated output.

So go on - what are you waiting for? Give it a try!