

XMLdb – V3.5 – NorthWind Paging Example

Introduction

Since the release of version V1.4 of XMLdb, support has been added to allow output paging. XMLdb cannot do this alone and requires some help from both SQL and XSL.

The associated SQL Stored Procedure must be able to handle paged requests. Specifically, the ability to return sections of a given result, so that the web page can display a reasonable amount of data.

Accordingly, the XSL must support the paging by providing links to other pages within the expected results. Typically a link to the “next” or “prev” page.

Using the NorthWind Example database that comes with SQL Server and specifically the Customers table that contains the names of all the Customers used in the Northwind examples, I have developed a paged document example that illustrates the type of SQL and XSL support needed.

The following screen shows the XMLdb module setup for this example.

XMLdb Settings

XML Data Source:

SQL Command: `exec XMLdb_PageXMLCustomers [page], 10`

Connection String: @northwind

Prefix: <root>

Suffix: </root>

Query String Checking: 30 **Maximum Paramter Size**

- Numeric Only Parameters**
- Enable SQL Injection Filtering**

Enable IMC:

Test Query String: page=1

Test Command **Add SQL Command** **Add CR/LF**

XSL Transformation Source:

Link Type:

- URL (A Link To An External Resource)**
- File (A File On Your Site)**

File Location: Root

File Name: NorthWind_Customers.xsl

[Upload New File](#)

Display Options (for Testing):

- Display any Parameters passed In**
- Show XSL source (after substitution)**
- Display SQL Command as Executed**
- Display XML as returned from SQL Command**

[Update](#) [Cancel](#)

Which generates a page like the following

CustomerID	CompanyName	ContactName	ContactTitle	Address	City	PostalCode	Country	Phone	Fax
ALFKI	Alfreds Futterkiste	Maria Anders	Sales Representative	Obere Str. 57	Berlin	12209	Germany	030-0074321	030-0076545
ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo	Owner	Avda. de la Constitución 2222	México D.F.	05021	Mexico	(5) 555-4729	(5) 555-3745
ANTON	Antonio Moreno Taquería	Antonio Moreno	Owner	Mataderos 2312	México D.F.	05023	Mexico	(5) 555-3932	
AROUT	Around the Horn	Thomas Hardy	Sales Representative	120 Hanover Sq.	London	WA1 1DP	UK	(171) 555-7788	(171) 555-6750
BERGS	Berglunds snabbköp	Christina Berglund	Order Administrator	Berguvsvägen 8	Luleå	S-958 22	Sweden	0921-12 34 65	0921-12 34 67
BLAUS	Blauer See Delikatessen	Hanna Moos	Sales Representative	Forsterstr. 57	Mannheim	68306	Germany	0621-08460	0621-08924
BLONP	Blondesd's père et fils	Frédérique Citeaux	Marketing Manager	24, place Kléber	Strasbourg	67000	France	88.60.15.31	88.60.15.32
BOLID	Bólido Comidas preparadas	Martin Sommer	Owner	C/ Araquil, 67	Madrid	28023	Spain	(91) 555 22 82	(91) 555 91 99
BONAP	Bon app'	Laurence Lebihan	Owner	12, rue des Bouchers	Marseille	13008	France	91.24.45.40	91.24.45.41
BOTTM	Bottom-Dollar Markets	Elizabeth Lincoln	Accounting Manager	23 Tsawassen Blvd.	Tsawassen	T2F 8M4	Canada	(604) 555-4729	(604) 555-3745

10 Records (1 to 10) [\[First\]](#) Page 1 of 10 [\[Next\]](#) [\[Last\]](#)

Step 1. SQL Stored Procedure

To be able to page through a result, a Stored Procedure is required that can be executed multiple times varying only the required page and return a consistent results.

There are two basic parameters needed: the page number, and the size of the page. The follow procedure is used in this example to page through the Customer Table.

```

create proc XMLdb_PageXMLCustomers
    @page int = 1,
    @size int = 20
as

declare @start int
declare @totalpages int
declare @count as int
declare @sql varchar(500)

set @count = (select Count(CustomerID) from    dbo.Customers )

set @start = (@page - 1) * @size
set @totalpages = (@count + (@size - 1)) / @size

select '<paging>' +
    '<page>' + cast(@page as varchar) + '</page>' +
    '<size>' + cast(@size as varchar) + '</size>' +
    '<totalrecs>' + cast(@Count as varchar) + '</totalrecs>' +
    '<totalpages>' + cast(@totalpages as varchar) + '</totalpages>' +
    '</paging>'

set @sql = 'select top ' + cast(@size as varchar(8)) + ' *
    from dbo.Customers
    where CustomerID not in
        ( select    top ' + cast(@start as varchar(8)) + ' CustomerID
          from dbo.Customers
          order by CustomerID
        )
    order by CustomerID
    for xml auto, elements'

    --print @sql
exec (@sql)

return
go

```

The procedure defines two input variables, page and size. For this example, the page size will be 20 as you can see in the module setup screen.

The procedure determines some paging information that will be passed to the XSL for processing as a XML fragment call "paging". This will be in the first select result generated by this procedure.

The procedure builds the second select as a string since the "Top N" records will be returned. Unfortunately, T-SQL does not allow the "Top N" expression to accept a variable number for "N", so we must generate the Select statement manually, and then execute it. The select uses a sub-select

to list all the values not wanted. *It is important that you order both the select and the sub-select in the same matter, or you will end up with weird results.*

Lastly, the generated select must output XML so the "FOR XML" option is required. In this example we also want the results as explicit elements (instead of attributes).

When the store procedure is executed as

```
EXEC XMLdb_PageXMLCustomer 3, 10
```

Results follow: (note the <root> element is added by XMLdb)

```
<root>
  <paging>
    <page>3</page>
    <size>10</size>
    <totalrecs>91</totalrecs>
    <totalpages>10</totalpages>
  </paging>
  <dbo.Customers>
    <CustomerID>FAMIA</CustomerID>
    <CompanyName>Familia Arquibaldo</CompanyName>
    <ContactName>Aria Cruz</ContactName>
    <ContactTitle>Marketing Assistant</ContactTitle>
    <Address>Rua Orós, 92</Address>
    <City>Sao Paulo</City>
    <Region>SP</Region>
    <PostalCode>05442-030</PostalCode>
    <Country>Brazil</Country>
    <Phone>(11) 555-9857</Phone>
  </dbo.Customers>
  ...
  <dbo.Customers>
    <CustomerID>GODOS</CustomerID>
    <CompanyName>Godos Cocina Típica</CompanyName>
    <ContactName>José Pedro Freyre</ContactName>
    <ContactTitle>Sales Manager</ContactTitle>
    <Address>C/ Romero, 33</Address>
    <City>Sevilla</City>
    <PostalCode>41101</PostalCode>
    <Country>Spain</Country>
    <Phone>(95) 555 82 82</Phone>
  </dbo.Customers>
</root>
```

Step 2. XSL Translation Document

The XSL used in this example is given below (file is part of the example files supplied)

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="html" version="4.0" encoding="windows-1252"/>
  <!-- extracted variables -->

  <xsl:variable name="size" select="root/paging/size"/>
  <xsl:variable name="page" select="root/paging/page"/>
  <xsl:variable name="totalrecs" select="root/paging/totalrecs"/>
  <xsl:variable name="lastpage" select="root/paging/totalpages"/>
  <!-- calculated variables -->

  <xsl:variable name="count" select="count(root/dbo.Customers)"/>
  <xsl:variable name="next" select="number($page) + 1"/>
  <xsl:variable name="prev" select="number($page) - 1"/>
  <xsl:variable name="start" select="((number($page) - 1) * number($size) + 1)"/>
  <xsl:variable name="end" select="($start - 1) + number($count)"/>

  <xsl:template match="paging">
    <!-- eats the paging data -->
  </xsl:template>

  <xsl:template match="/">
  <table width="100%" border="1">
    <tr>
      <th>CustomerID</th>
      <th>CompanyName</th>
      <th>ContactName</th>
      <th>ContactTitle</th>
      <th>Address</th>
      <th>City</th>
      <th>PostalCode</th>
      <th>Country</th>
      <th>Phone</th>
      <th>Fax</th>
    </tr>
    <xsl:apply-templates/>
  </table>
  <table width="100%" border="0">
    <tr>
      <td align="left" width="50%">
        <xsl:value-of select="$count"/> Records
        (<xsl:value-of select="$start"/> to <xsl:value-of select="$end"/>)
      </td>
      <td align="right" width="50%">
        <xsl:choose>
          <xsl:when test="$lastpage > 2">&#160;
            <a
              <xsl:attribute name="href">
                [CurrentPageUrl]?page=1
              </xsl:attribute>[First]
            </a>
          </xsl:when>
        </xsl:choose>
        <xsl:choose>
          <xsl:when test="$page > 1">&#160;
            <a
              <xsl:attribute name="href">
                [CurrentPageUrl]?page=<xsl:value-of
                  select="$prev"/></xsl:attribute>[Prev]
            </a>
          </xsl:when>
        </xsl:choose>
        Page <xsl:value-of select="$page"/> of <xsl:value-of select="$lastpage"/>
        <xsl:choose>
          <xsl:when test="$next &lt;= $lastpage">&#160;
            <a
              <xsl:attribute name="href">
                [CurrentPageUrl]?page=<xsl:value-of
                  select="$next"/></xsl:attribute>[Next]
            </a>
          </xsl:when>
        </xsl:choose>
      </td>
    </tr>
  </table>
  </xsl:template>
</xsl:stylesheet>
```

```

        <xsl:when test="$lastpage > 2 and $page &lt; $lastpage ">&#160;
            <a>
                <xsl:attribute name="href">
                    [CurrentPageUrl]?page=<xsl:value-of
                        select="$lastpage "/></xsl:attribute>[Last]
                </a>
            </xsl:when>
        </xsl:choose>
    </td>
</tr>
</table>
</xsl:template>

<xsl:template match="dbo.Customers">
    <tr>
        <td bgcolor="lightgrey" width="20%">
            <b><xsl:value-of select="./CustomerID"/></b>
        </td>
        <td width="10%" align="center">
            <xsl:value-of select="./CompanyName"/>
        </td>
        <td align="center">
            <xsl:value-of select="./ContactName"/>
        </td>
        <td align="center">
            <xsl:value-of select="./ContactTitle"/>
        </td>
        <td align="center">
            <xsl:value-of select="./Address"/>
        </td>
        <td align="center">
            <xsl:value-of select="./City"/>
        </td>
        <td align="center">
            <xsl:value-of select="./PostalCode"/>
        </td>
        <td align="center">
            <xsl:value-of select="./Country"/>
        </td>
        <td align="center">
            <xsl:value-of select="./Phone"/>
        </td>
        <td align="center">
            <xsl:value-of select="./Fax"/>
        </td>
    </tr>
</xsl:template>
</xsl:stylesheet>

```

This tutorial is not about how XSL Stylesheet works, but how XMLdb works with the results of the translation. However, you may want to study this XSL to determine how the processing occurs.

Several items to note in this XSL:

- The first Select Result in the Store Procedure is used to define paging elements within the StyleSheet. A Template is used to match the entire node structure and stop any associated output when found in the input stream.
- Within the "When" clauses the condition be tested requires the less-than symbol, but XSL will consider this illegal and generate an error, so "<" is used instead of the "<". XSL processes this with out error.
- "[CurrentPageUrl]?tabid=[tabID]" exists in the output. XMLdb will substitute the appropriate values for both [CurrentPageUrl] and [TabID]. This allows the page name and tab to changes without having to alter the XSL accordingly.


Step 3. Defining XML Module.

As the example screen shot shows:

The screenshot shows the XMLdb Settings dialog box. The 'XML Data Source' section is expanded, showing the following configuration:

- SQL Command:** `exec XMLdb_PageXMLCustomers [page], 10`
- Connection String:** `@northwind`
- Prefix:** `<root>`
- Suffix:** `</root>`
- Query String Checking:** 30 (Maximum Parameter Size)
- Numeric Only Parameters
- Enable SQL Injection Filtering
- Enable IMC
- Test Query String:** `page=1`
-
- Add SQL Command
- Add CR/LF

1. The Stored Procedure is used in the SQL Command. The page value is defined as a substitution for a Query Variable "Page".
2. Maximum Parameter Size is defaulted to 30.
3. SQL Injection Filtering is enabled by default, however in this example I have disabled the filtering.
4. A Default Query String Variable is defined as "Page=1". When the page is loaded for the first time, no "Page" variable will exist, so the value of 1 will be used to change the SQL Command to: "exec XMLdb_PageXMLCustomer 1, 20"

 XSL Transformation Source:

Link Type:

URL (A Link To An External Resource)

File (A File On Your Site)

File Location:

Root ▼

File Name:

NorthWind_Customers.xsl ▼

[Upload New File](#)

5. The internal file "NorthWind_Customers.xsl" has been selected as XSL Transformation file. The file must be download via the File Manager for this portal.

Step 4. Execution

When XMLdb module is executed, the saved SQL command is retrieved from the module settings table. Any supplied Query Variables are substituted against the SQL Command, then the default Query Variables are substituted. Note, that if a supplied variable is substituted, the target string ("[variable]") will have been removed from the SQL command, so the default Query Variables will not find any target string.

Once the SQL Command has been executed, the XSL file is read and also is substituted first for supplied variables, then default variables.

Finally, the XSL Translator is handed the XML and XSL data for processing. The resulting HTML output is then inserted into the current page being rendered by the DNN framework.

The result should look something like the following:

CustomerID	CompanyName	ContactName	ContactTitle	Address	City	PostalCode	Country	Phone	Fax
ALFKI	Alfreds Futterkiste	Maria Anders	Sales Representative	Obere Str. 57	Berlin	12209	Germany	030-0074321	030-0076545
ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo	Owner	Avda. de la Constitución 2222	México D.F.	05021	Mexico	(5) 555-4729	(5) 555-3745
ANTON	Antonio Moreno Taquería	Antonio Moreno	Owner	Mataderos 2312	México D.F.	05023	Mexico	(5) 555-3932	
AROUT	Around the Horn	Thomas Hardy	Sales Representative	120 Hanover Sq.	London	WA1 1DP	UK	(171) 555-7788	(171) 555-6750
BERGS	Berglunds snabbköp	Christina Berglund	Order Administrator	Berguvsvägen 8	Luleå	S-958 22	Sweden	0921-12 34 65	0921-12 34 67
BLAUS	Blauer See Delikatessen	Hanna Moos	Sales Representative	Forsterstr. 57	Mannheim	68306	Germany	0621-08460	0621-08924
BLONP	Blondesddsl père et fils	Frédérique Citeaux	Marketing Manager	24, place Kléber	Strasbourg	67000	France	88.60.15.31	88.60.15.32
BOLID	Bólido Comidas preparadas	Martín Sommer	Owner	C/ Araquil, 67	Madrid	28023	Spain	(91) 555 22 82	(91) 555 91 99
BONAP	Bon app'	Laurence Lebihan	Owner	12, rue des Bouchers	Marseille	13008	France	91.24.45.40	91.24.45.41
BOTTM	Bottom-Dollar Markets	Elizabeth Lincoln	Accounting Manager	23 Tsawassen Blvd.	Tsawassen	T2F 8M4	Canada	(604) 555-4729	(604) 555-3745

10 Records (1 to 10) [First] Page 1 of 10 [Next] [Last]

The bottom line displays the number of records, their relative position, a link to the first page, a link to the previous page, the current page out of total number of pages, a link to the next page, and a link to the last page.

The link passed a new value back for the page requested on the postback of the page.

Files Included

XMLdb_PageXMLCustomer.sql – the stored procedure to add to the DotNetNuke database. Load the file into Query Analyzer and execute. This will drop the procedure if found and reload a new version.

If you do not have Microsoft's Query Analyzer, you can load the procedure using isql as follows:

```
Isql -S 127.0.0.1 -U username -P password -d Northwind -n -i XMLdb_PageXMLCustomer.sql
```

Where username, password are the appropriate values for your database. This command assumes your database is located on the local machine, otherwise use the appropriate computer name or IP address.

NorthWind_Customers.xsl – the XSL style sheet shown in step 3. This file must be downloaded into the portal files via the DNN Admin File Manager.

Alternatively, copy the file into the appropriate portal directory and used the File Manager to re-sync the files.

content.XMLdb.XMLdbPagingExample.xml – the DNN import file for XMLdb. This file must be download into the portal files to be used. Once XMLdb module has been placed on a page, use the Import function to import this file.

Please note that the file contains the a Connection string to access the Northwind database. The database is assume to be located on the (local) server and required a userid and password. You may edit is file and replace the ??? with appropriate information. Alternatively, you can wait to correct this information via the XMLdb Settings page once the import is complete.