Paper 38-26

Using ORACLE XSQL Servlet to Access a SAS Database: Simplifying the XML Development Process

Michael W. Deines, Preferred Consulting, Inc., Kendall Park, NJ

ABSTRACT

Using ORACLE XSQL Servlet, a SAS database can be accessed and output can be presented in an XML format. As part of ORACLE XSQL Servlet package, an XML parser for Java, XML Query Utility, and XSLT processor are provided. Adding the SAS JDBC driver information into ORACLE XSQL Servlet JDBC driver XML document allows the servlet to access SAS data maintained by a SAS/SHARE server. Query results are then stored in an XML document, presented back to the browser. The example application also includes an interface that allows users to load a WORD document onto the web server and extract information from the document into a XSL file. This file will be reference by the ORACLE XSQL Servlet output and apply conditional processing to display HTML information specific for the results of the query.

The power of XML

Accessing data on-line and presenting the information in a manner that best suits the individual and/or machine is the power of XML. That is why XML has been received with open arms throughout the community. Combining XML and XSL provides a powerful interface between various sources of data: imbedded format information, documented data source within the data itself, and flexible data references are a few of the benefits.

Streamlining XML

ORACLE XSQL Servlet packages the functionality of parsing of an XML document, the creation of an API to access the contents of an XML document, and an XSLT processor to process the data extracted from an ORACLE database. The servlet can also be modified to access SAS data being served by a SAS/SHARE server session, thus providing packaged interface that generate XML structured data from SAS databases.

Setting up ORACLE XSQL Servlet to access a SAS/SHARE Session

Make sure that either of the following JDKs were installed:

JDK 1.1.8
JDK 1.2.2

Make sure that one of the following servlet engines have been installed:

Oracle Internet Application Server 8*i*
Allaire JRun 2.3.3
Apache 1.3.9 with JServ 1.0 and 1.1
Apache 1.3.9 with Tomcat 3.1 Servlet Engine
Apache Tomcat 3.1 Web Server + Servlet Engine
NewAtlanta ServletExec 2.2 for IIS/PWS 4.0
Oracle8*i* Lite Web-to-Go Server
Oracle8*i* 8.1.7 Oracle Servlet Engine

Sun JavaServer Web Development Kit (JSWDK) 1.0.1 Web Server

Development for this application was done on NewAtlanta ServletExec 2.2 for IIS/PWS 4.0.

Software and installation instructions are available for ORACLE XSQL Servlet is available on ORACLE's web site (www.oracle.com).

Once the ORACLE XSQL Servlet has been set up correctly, the JDBC for SAS can be defined.

Setting up the JDBC connections to a SAS/SHARE server

ORACLE XSQL Servlet uses an XML file to obtain the connection parameters needed to access a SAS/SHARE server. This file is located under the ".\xsql\lib\XSQLConfig.xml" where "." represents the directory the directory below the "xsql" unpackage directory.

Structure of JDBC connection XML file:

```
<?xml version="1.0" ?>
<XSQLConfig>
  <connectiondefs>
    <connection name="testdata">
      <username></username>
      <password></password>

<dburl>jdbc:oracle:thin:@localhost:1521:ORCL</db
url>
      <driver>oracle.jdbc.driver.OracleDriver</driver>
    </connection>
    <connection name="testlite">
      <username>litetest</username>
      <password>litetest</password>
      <dburl>jdbc:Polite:POlite</dburl>

<driver>oracle.lite.poljdbc.POLJDBCDriver</driver>
    </connection>
  </connectiondefs>
</XSQLConfig>
```

For the servlet to recognize the SAS/SHARE server, a connection defintion must be created.

Defining a connection

1) Copy and paste one “connection” node under “connectiondefs” node
2) Rename new “connection” node (name=”SASJDBC”)
3) Enter username between “username” tags under new “connection” node (no quotes)
4) Enter password between “password” tags under new “connection” node (no quotes)
5) For “dburl” tags enter “jdbc:sharenet:@machine:port” (no quotes)
6) For “driver” tags enter “com.sas.net.sharenet.ShareNetDriver” (no quotes)
7) Save XML file

Accessing SAS data

Developers can define SAS/SHARE libraries when starting the SAS/SHARE server before the PROC SERVER procedure.

An XSQL file, a file with an “xsql” extension, is the file type that will be recognized by the servlet.

Structure for XSQL file:

```
<?xml version="1.0"?>
<xsql:query xmlns:xsql="urn:oracle-xsql"
connection="SASJDBC">
   select b.question_name,
          c.answer_name
   from  online.test_it1 a,
          online.test_it2 b,
          online.test_it3 c
  where   a.question_id=b.question_id and
          a.question_id=c.question_id and
          a.answer_id=c.answer_id
</xsql:query>
```

Between the “xsql:query” tags is the query that will be extracting data from the SAS/SHARE library. The attribute “xmlns:xsql” the namespace prefix “xsql” and the name of the namespace it represents “urn:oracle-xsql”. The connection attribute contains the name of the connection that will reference the SAS/SHARE server.

After this file is saved, there is one more step that must be done before the servlet can present the output from the SAS/SHARE session. When the servlet tries to present the SAS data in XML format, it tries to use the SAS labels as the data tags in the output. If a SAS data set does not have labels for the columns, the browser will display an array out-of-bounds error. To fix this problem, simply provide labels to the SAS data sets on the SAS/SHARE server.

The output to the browser of the XSQL extract looks like this:

```
<?xml version="1.0" ?>
- <ROWSET>
  - <ROW num="1">
      <question_name>QUESTION1
          </question_name>

          <answer_name>UnMarked</an
          swer_name>
    </ROW>
  - <ROW num="2">
      <question_name>QUESTION2
          </question_name>

          <answer_name>Marked</answ
          er_name>
    </ROW>
  - <ROW num="3">
      <question_name> QUESTION3
          </question_name>

          <answer_name>GOOD</answer
          _name>
    </ROW>
  - <ROW num="4">
      <question_name> QUESTION4
          </question_name>

          <answer_name>FAIR</answer_
          name>
    </ROW>
  - <ROW num="5">
      <question_name> QUESTION5
          </question_name>

          <answer_name>FAIR</answer_
          name>
    </ROW>
</ROWSET>
```

The tag “ROWSET” contains all the rows from the extract.
The each tag “ROW” contains one row of data.
The attribute “num” of tag “ROW” contains the row number.
In each “ROW” tags, tag pairs for each column of the extract.
In each tag pair for each column is the value of the extracted column.

The application has presented the extracted information in an XML format that can be used anywhere by another application. For this example, business rules are going to be applied such that an appropriate output is displayed to browser.

The XSL Layer

A separate process was set up by which an analyst uploaded a WORD document with business logic for each desired output. The WORD document is parsed in EXCEL using an EXCEL macro and saved as an EXCEL spreadsheet. SAS then processes the EXCEL spreadsheet to create an XSL file. Developers can come up with their own process for creating the XSL.

The XSL file is as follows:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
    <HTML>
    <HEAD/>
    <BODY>
    <p>
<xsl:for-each select="/ROWSET">
<xsl:if test='//question_name[. =
"QUESTION1"]/../answer_name="Marked"
or
//question_name[. =
"QUESTION2"]/../answer_name="Marked" '>
<a href="Q1Q2">Question 1 and Question 2 are
marked</a><br/>
</xsl:if>
</xsl:for-each>
    </p>
    </BODY>
    </HTML>
</xsl:template>
</xsl:stylesheet>
```

Between the “xsl:stylesheet” tags contains the logic to test and display HTML information. The attribute “xmlns:xsl” the namespace prefix “xsl” and the name of the namespace it represents http://www.w3.org/1999/XSL/Transform which helps identify the XSLT elements. The attribute “xsl:stylesheet” specifies the xsl file is a style sheet. The tag “xsl:template” describes a structure to be applied on an XML. This template looks for the root node “/”, described by match attribute of the “xsl:template” node.

When the root node is located:

The first output is the html tags <HTML>,<HEAD/>,<BODY> and <p>.

The “xsl:for-each” tag runs code in “xsl:for-each” tag for each occurrence of the select statement

For each “ROWSET”:

The “xsl:if” tags checks the Boolean value of the “test” attribute. If the “test” attribute is “true” then process what is in-between the tags.
In this example, if QUESTION1=”Marked” or QUESTION2=”Marked” then the link will be sent to the browser. Note that “//question_name” means get the value of “question_name” from wherever located above the “ROWSET” node.

Applying the XSL on the XML would result in a HTML document sent to the browser. All the XML processing would be performed on the server.

Notice the power and flexibility of XSL and XML. Even though the data that needed to be compared exist on different rows, XSL was still able to locate the information without changing the structure of the XML document.

CONCLUSION

ORACLE XSQL Servlet provides the ability create XML documents from SAS data maintained by a SAS/SHARE server. XSL files allow developers to write business logic that can be applied on the output XML files such that users will view results dependent upon business rules.

ACKNOWLEDGMENTS

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Michael W. Deines
Preferred Consulting, Inc.
5 Drexel Hill Drive
Kendall Park NJ 08824
Work Phone: (732) 951-1285
Fax: (732) 951-1285
Email: mdeines@home.com