CacheTables ESB Service

PSDN Sonic Code Share Contest Entry

Submitted by: Jon Nyholt (Solutions Architect)

Company: Boehringer Ingelheim Canada Ltd.

Date: May 30, 2008

Summary

The CacheTables ESB service provides a framework for storing frequently used data in a memory resident database embedded in a host MF Container. The typical configuration for using the CacheTables service is outlined in the diagram below. Data can be stored and updated in the database through the entry endpoint of the service. The contents of the database can then be queried, and updated from XML Transformation services deployed within the same container, using “query” and “update” helper methods.

There are several advantages to using the CacheTables service:

- The CacheTables service allows a Sonic ESB developer to build database lookups into an XSLT stylesheet without making database calls over the network. Since the database is stored in memory within the container the lookups are very efficient.
- It also allows a developer to aggregate lookup data from several source systems into one data store. In addition, it allows a developer to match the table names and column names to a canonical format which will help to simplify the lookup logic used in the XSLT stylesheet.

The following diagram outlines the basic functionality of the CacheTables service:

The CacheTables database can be configured to use one of multiple possible open-source embedded Java databases. The H2 Database, HSQLDB Database, and Apache Derby Database are all possible options. (The H2 database is configured as the default database.) The database driver name and connection string are provided as initialization parameters of the service. All the embedded Java databases mentioned automatically persist their data to files, and the CacheTables service inherits this functionality and persists and restores data when a container is shutdown and restarted.

Instructions

The following steps outline the standard usage of the CacheTables service.

1) Deploy the CacheTables service to an ESB container and specify all initialization parameters. The necessary database library (jar file) will need to be included in the container classpath. (The latest versions of H2, HSQLDB, and Derby libraries are available in the CacheTables lib folder.)

2) Once the CacheTables service is deployed to an ESB container, restart the container and check the container log file to ensure that the service and database initialized successfully.

```
Sonic ESB Professional Developer Edition [Serial Number: 999999999]
Release 7.0.2 Build Number 172
Copyright (c) 1999-2006, Sonic Software Corporation.
All rights reserved.

[08/05/29 17:21:11] ID=dev_ESBTest (info) [ESBService] Initializing service dev.Transform
[08/05/29 17:21:11] ID=dev_ESBTest (info) [ESBService] Service dev.Transform initialized: 1 thread(s) listening on Endpoint dev.Transform.Entry
[08/05/29 17:21:11] ID=dev_ESBTest (info) [ESBService] Initializing service dev.CacheTables
[08/05/29 17:21:11] ID=dev_ESBTest (info) [ESBService] Service dev.CacheTables initialized: 1 thread(s) listening on Endpoint dev.CacheTables.Entry
[08/05/29 17:21:11] ID=dev_ESBTest (info) [dev.CacheTables] Initializing ...
[08/05/29 17:21:11] ID=dev_ESBTest (info) [dev.CacheTables] Initializing the database ...
[08/05/29 17:21:13] ID=dev_ESBTest (info) Looking Glass instrumentation is disabled.
[08/05/29 17:21:13] ID=dev_ESBTest (info) Looking Glass stabilizers not enabled.
[08/05/29 17:21:13] (info) Loaded ID=dev_ESBTest
[08/05/29 17:21:13] ID=dev_ESBTest (info) Container's ESB (JMS) Connection: jms_defaultConnection
[08/05/29 17:21:13] ID=dev_ESBTest (info) Container's HTTP Routing Connection: http_defaultConnection
[08/05/29 17:21:13] (info) ...startup complete
```

3) To populate the database with data submit a properly formatted <CacheTables> XML message to the entry endpoint of the CacheTables service. (A sample message is provided in the CacheTables test folder, and the schema is provided in the xsd folder.)

   The <CacheTables> XML message contains two branches: <UpdateTables> and <DumpContents>.

   The <UpdateTables> branch contains a series of <Statement> elements which contain typical SQL statements. The SQL statements are executed against the CacheTables database in sequential order.

   The <DumpContents> branch contains a series of <Table> elements which contain the names of tables in the database. For each <Table> element included the CacheTables service will dump the table contents and add it as an additional message part of the outbound message.

   A sample CacheTables input message is provided:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<CacheTables xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="C:\Workspace\Sonic\Utility_CacheTables\xsd\CacheTables.xsd">
    <UpdateTables>
        <Statement>
            DROP TABLE Items
        </Statement>
        <Statement>
            CREATE TABLE Items
            (ItemNum INT PRIMARY KEY, Name VARCHAR(32), Barcode VARCHAR(16), DiscountPercent DOUBLE)
        </Statement>
        <!-- MERGE statement is available in H2 database, but not HSQLDB or Derby -->
        <Statement>
            MERGE INTO Items (ItemNum, Name, Barcode, DiscountPercent)
            KEY (ItemNum) VALUES (100, 'Blue Widget', '1234567890', '0.00')
        </Statement>
        <Statement>
            MERGE INTO Items (ItemNum, Name, Barcode, DiscountPercent)
            KEY (ItemNum) VALUES (101, 'Green Widget', '1234567891', '0.00')
        </Statement>
        <Statement>
            MERGE INTO Items (ItemNum, Name, Barcode, DiscountPercent)
            KEY (ItemNum) VALUES (102, 'Red Widget', '1234567892', '0.00')
        </Statement>
        <Statement>
            MERGE INTO Items (ItemNum, Name, Barcode, DiscountPercent)
            KEY (ItemNum) VALUES (103, 'Orange Widget', '1234567893', '0.10')
        </Statement>
    </UpdateTables>
    <DumpContents>
        <Table>Items</Table>
    </DumpContents>
</CacheTables>
```

When a message is submitted to the entry endpoint of the CacheTables service the results of the execute statements will be logged in the container log file. Check the log file to ensure all update statements were executed successfully. (If container debugging messages are enabled additional details will be provided.)

```
[08/05/29 19:22:20] ID=dev_ESBTest (info) [dev.CacheTables] Executing UpdateTables statements ...

[08/05/29 19:22:20] ID=dev_ESBTest (info) [dev.CacheTables] 6 update statement(s) executed.

[08/05/29 19:22:20] ID=dev_ESBTest (info) [dev.CacheTables] 0 invalid statements(s) encountered.
```

Once data has been populated within the CacheTables database it can be queried from an XML Transformation service using the “query” and “update” XSLT helper methods. (The helper methods are similar to the Sonic ESB header extensions.) In order to use the helper methods within an XSLT stylesheet include the xmlns:cache="java:com.psdn.esb.service.Database" attribute in the stylesheet element. The syntax for the helper methods are cache:query(“query string”) and cache:update(“update string”).

The query string provided in the query method is a standard SQL select statement. The method will return a string representation of the first row, first column of the result set. If no records are found, or if the statement doesn’t execute correctly, an error message will be returned.

A sample stylesheet is provided:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:cache="java:com.psdn.esb.service.Database" exclude-result-prefixes="cache">

    <xsl:template match="node() | @*">
        <xsl:copy>
            <xsl:apply-templates select="@* | node()"/>
        </xsl:copy>
    </xsl:template>

    <xsl:template match="Barcode">
        <xsl:variable name="lookupBarcode">
            select Barcode from Items
            where ItemNum = '<xsl:value-of select="../ItemNum"/>'
        </xsl:variable>
        <xsl:variable name="Barcode"><xsl:value-of select="cache:query($lookupBarcode)"/></xsl:variable>

        <Barcode>
            <xsl:value-of select="$Barcode"/>
        </Barcode>
    </xsl:template>

    <xsl:template match="SalePrice">
        <xsl:variable name="lookupDiscountPercent">
            select DiscountPercent from Items
            where ItemNum = '<xsl:value-of select="../ItemNum"/>'
        </xsl:variable>

        <xsl:variable name="DiscountPercent"><xsl:value-of select="cache:query($lookupDiscountPercent)"/></xsl:variable>

        <SalePrice>
            <xsl:value-of select="format-number(../UnitPrice - (../UnitPrice * $DiscountPercent),'#.00')"/>
        </SalePrice>
    </xsl:template>

</xsl:stylesheet>
```

A sample XML message to run against the stylesheet is also provided:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Order>
    <Customer>
        <CustNum>12345</CustNum>
        <Name>Widget Buyers Incorporated</Name>
    </Customer>
    <OrderLines>
        <Line>
            <ItemNum>100</ItemNum>
            <Name>Blue Widget</Name>
            <Barcode></Barcode>
            <Quantity>25</Quantity>
            <UnitPrice>65.99</UnitPrice>
            <SalePrice></SalePrice>
        </Line>
        <Line>
            <ItemNum>101</ItemNum>
            <Name>Green Widget</Name>
            <Barcode></Barcode>
            <Quantity>10</Quantity>
            <UnitPrice>44.99</UnitPrice>
            <SalePrice></SalePrice>
        </Line>
        <Line>
            <ItemNum>103</ItemNum>
            <Name>Orange Widget</Name>
            <Barcode></Barcode>
            <Quantity>15</Quantity>
            <UnitPrice>72.99</UnitPrice>
            <SalePrice></SalePrice>
        </Line>
    </OrderLines>
</Order>
```

When the sample message is processed with an XML Transformation service, using the sample stylesheet, the CacheTables database is referenced to populate the missing values.

The completed output message is shown below:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Order>
    <Customer>
        <CustNum>12345</CustNum>
        <Name>Widget Buyers Incorporated</Name>
    </Customer>
    <OrderLines>
        <Line>
            <ItemNum>100</ItemNum>
            <Name>Blue Widget</Name>
            <Barcode>1234567890</Barcode>
            <Quantity>25</Quantity>
            <UnitPrice>65.99</UnitPrice>
            <SalePrice>65.99</SalePrice>
        </Line>
        <Line>
            <ItemNum>101</ItemNum>
            <Name>Green Widget</Name>
            <Barcode>1234567891</Barcode>
            <Quantity>10</Quantity>
            <UnitPrice>44.99</UnitPrice>
            <SalePrice>44.99</SalePrice>
        </Line>
        <Line>
            <ItemNum>103</ItemNum>
            <Name>Orange Widget</Name>
            <Barcode>1234567893</Barcode>
            <Quantity>15</Quantity>
            <UnitPrice>72.99</UnitPrice>
            <SalePrice>65.69</SalePrice>
        </Line>
    </OrderLines>
</Order>
```

4) If it is necessary to do so, the “update” helper method can be used in much the same way as the “query” method to update the contents of the CacheTables database using SQL update, merge, and delete statements. However, this functionality can be disabled by setting the CacheTables enableUpdateHelper initialization parameter to “false”.