Paper 203-28

Developing SAS-Ready Analyzable Data Systems A Java Web Application for Creation and Management of SAS Relational Databases

John R. Copeland, David W. Walker, David W. King
Statistical Analysis Branch, Data Management Division, National Immunization Program, Centers for Disease Control and Prevention

ABSTRACT

When data are collected at CDC, non-SAS file types are often used for initial storage. Collecting data in one of these formats requires the analyst to execute extra steps, to transform the data into a format suitable for SAS analysis procedures. These extra steps may be eliminated if a database consisting of SAS data tables is created before data collection, and data are stored in this SAS database as they are collected. However, those responsible for data collection often lack the SAS knowledge necessary for creating a SAS database and must therefore use other tools for data collection. This presentation describes a Java Web application that provides a solution to this problem. The Java Web application consists of Java Servlets, Java Server Pages (JSPs), JavaBeans, and Java Database Connectivity (JDBC). This application assigns a directory on its host server to each database created and uses JDBC to create tables and to define columns and column constraints, displaying the metadata as the tables are created. Web applications are already improving the efficiency of data collection, management, and analysis at the CDC. This Java Web application can make it easier for investigators to collect data in analyzable SAS format.

BACKGROUND

SAS is a tool used widely for data analysis at the Centers for Disease Control and Prevention (CDC). When data are collected, however, other file types and software applications are often used. Historically, these other file types have consisted of delimited text files or other proprietary formats such as spreadsheets or databases. When data are collected and stored in these formats, there are often tedious intermediate steps (conversion of files into SAS data tables, or establishing connections to relational databases) that must be taken by a data manager or analyst before SAS computations or analysis procedures may be executed. Lately, the trend at CDC has been creating Web forms for data entry and storing the data in non-SAS relational databases. This trend has served to make more efficient the tasks of data entry and management. But still more efficiency could be realized if the data were to be stored in a relational database consisting of SAS data tables, as soon as the data were collected. Unfortunately, those responsible for developing data storage applications are often trained in administering databases other than SAS, and do not know how to create empty relational SAS tables. This paper describes a Java Web application that serves as an administrator's tool for SAS databases. The application allows a database administrator (DBA) with limited SAS knowledge to create databases and add or remove SAS tables from those databases; defining, formatting, and labeling columns; and defining column constraints.

DESIGN

This Java Web application was developed with SAS AppDev Studio. It seeks to employ the model, view, controller (MVC) design pattern, with a Java Servlet as the controller, JSPs as the views, and JavaBeans as the models. The application runs on a Tomcat Web server on a Sun Microsystems Sunfire 6800 using the Solaris 8 platform. Its chief features are the execution of shell commands on the host server, and the employment of the SAS JDBC driver that ships with AppDev Studio.

SAS databases on this platform consist of SAS tables (datasets) stored in subdirectories within a parent directory called "sasdbs". Upon gaining access to this application (it is password-protected), a user may either begin a new database or work with an existing database. If the user chooses to add a new database, the user is prompted to give a name to this database and a Unix shell script is executed, assigning this name to a new subdirectory of sasdbs. When a user chooses to manage an existing database, existing SAS tables may be viewed or modified, or new tables may be created and added to the database directory. To create a new table, the user assigns a valid SAS name to the table, and adds columns, one-by-one, to the table. The user must give a valid SAS variable name to each column, and select either: CHAR, NUM, DATE, or TIMESTAMP as the data type for the column. (In SAS tables, of course, date and datetime variables are of the numeric type. This application assigns an appropriate format if the user wishes to define a DATE or TIMESTAMP variable.) The user also has the option of assigning the length or a label to each column. In addition to adding columns to a table, constraints on columns may also be defined. Optional constraints include: restricting the legal values for a column, declaring a column to reject null values, and assigning a single-column or complex primary key. As the user defines columns and constraints, the metadata are stored in *server* memory in Java Vector objects. The application then constructs the appropriate SQL "CREATE TABLE" statement from the information contained in the Vector objects. JDBC and SAS/Share are used to execute this statement, creating the SAS table in the desired directory. JDBC and SAS/Share are also used when an existing table is selected. In this case, the application displays the table's metadata, and gives the user the options of viewing the data in selected columns or modifying the table by adding or removing columns.

JDBC AND SAS/SHARE

The SAS JDBC driver that is shipped with AppDev Studio can execute SQL statements on a SAS/Share server. Starting a SAS/Share server on a Unix server is simple. A server name and port number are added to the /etc/services file. Typically, port 5010 is used for the SAS/Share service, and our service is named "shr1". Then a SAS batch process is started that includes the following procedure:

```
proc server id = shr1;
run;
```

The process may also contain **`libname`** statements if the administrator wishes to define permanent SAS libraries available to all SAS/Share users.

This Web application uses JDBC to establish a Java Connection to the shr1 service. The static method, "makeConnection", is part of a Java class called "AppUtilities" and is used to create this Connection:

```
package sasdata;

import java.io.*;
import java.util.*;
import java.net.*;
import java.sql.*;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class AppUtilities
{
  public static Connection makeConnection(
      HttpServletRequest request,
      HttpServletResponse response)
      throws Exception
  {
      Connection conn = null;
      String serverIP = "localhost";
      String serverPort = "5010";
      String dbLibPath = "/projects/sasdbs/" +
        (String) request.getSession().getAttribute(
          "dbName");
      String data = "jdbc:sharenet://" +
          serverIP + ":" + serverPort;
      Properties prop = new Properties();
      Object o = prop.setProperty("librefs",
          "mysaslib '" + dbLibPath + "'");
      try
      {
        Class.forName(
          "com.sas.net.sharenet.ShareNetDriver");
        com.sas.net.sharenet.ShareNetDriver snd =
          new com.sas.net.sharenet.ShareNetDriver();
        conn = snd.connect(data, prop);
      }  // End try
      catch (Exception e)
      {
        throw e;
      }  // End catch
      return conn;
  }  // End makeConnection method
}  // End AppUtilities class
```

Once established, this Connection is stored in server memory as a session attribute and may be repeatedly used for execution of SQL statements as long as the session is alive.

DISCUSSION

Collection of data has been made more efficient at the CDC by the use of relational databases employing Web applications for data entry. Web developers and DBAs are not often trained to work with SAS data tables, and so must often use other software for their data storage. This Java Web application offers DBAs another option – an option preferable to the statisticians who will be analyzing the data. With this application, a DBA needs no SAS-specific knowledge to create databases consisting of SAS tables. A Web developer may then use already-familiar methods for storing data in these SAS tables.

In addition to the application described here, Java and SAS may be used together to develop powerful data reporting and analysis tools. Using SAS data management and analysis power to create informative datasets and using the ability of Java to accept user input and conditionally query these datasets, Web applications may be developed which will deliver highly customized information to the requestor. SAS and Java are powerful tools that complement each other well. We hope to do more to explore the potential of this combination.

CONTACT INFORMATION

John Copeland can be reached by phone at 404-639-8866, or by email at JCopeland@cdc.gov.