Paper 115-2011

SAS® Flirts Java: Power Point Creation

Laiju Zhang, Medical Consultants Inc
49 Plain St. North Attleboro, MA 01772

Abstract

This paper presents the process for creating power point slides shows using SAS and Java. It describes how to create object from SAS DATA step by using the SAS recent addition of javaObject to its version 9, how to use this object created within SAS session to communicate with Java package apache poi hslf, thus creating power point slides shows. Further realizing the difficulties of communicating with external independent Java objects when creating and using Java objects within SAS DATA step, the paper explores the possibility of enhancing the power point creation process by letting SAS write Java programs that create the power points slides show. The net result is a coherent SAS program that both generates the necessary Java programs for utilizing Apache POI package for creating power point slides show, and all necessary SAS programs that prepare the information needed for power point generation, as well as the system level manipulation from within SAS for compiling the Java program, and for creating and utilizing Java objects from within SAS DATA step, thus making passing information from various SAS procedures to the Java facilities a seamless process.

Key words: power point, Java, macros.

Source code link

1. Introduction: from SAS world to Java World

SAS programmers live in SAS world. To live a better life even in SAS world (to be more powerful in programming activities), however, we need other worlds, Java, Perl, Python, R, etc. We not only need to command the tools of SQL, HASH, Perl Regular Expressions, R functions, we also need to learn the new comers that are recently introduced in SAS – Java object.

This paper is an attempt to describe my own journey of learning Java with the mindset of a SAS programmer.

It first explains how to use java object within SAS by providing a simple example of how to use SAS as a tool for Java. It then goes on to illustrate how to create power point presentation using Java's POI library. Finally, the paper explains how to enhance our programming power by taking Java as part of SAS – to use Java for SAS jobs.

2. SAS as Part of Java

SAS DATA step gives us the ability to communicate with external Java programs and therefore opens a whole different programming world. With the creation of Javaobj we can instantiate Java classes and access fields and methods on the resultant objects. Regardless of all kinds of computer terminologies, we

were almost working in the Java environment. Yes, we are still programming in SAS; but we must follow the rules of Java at this very moment. Therefore, it is very important that when we use java objects within SAS session, we understand what rules Java plays.

Perhaps for a SAS programmer with procedure programming habit, it is important to understand one basic rule of Java at least: "OOP" – object oriented programming. With Java, we start with writing a class which sets up a plan for creating objects by specifying data and behavior. The data refer to the fields, properties, variables, and their values; the behavior is realized by various kinds of methods. Classes plan the combination of data and behavior in terms of designs; objects realize them and materialize the plans in terms of implementations.

Take the following code from SAS site for example:
(http://support.sas.com/rnd/base/datastep/dot/javaobj.html)

```
import java.util.*;
import java.lang.*;
public class ttest
{
  public int    i;
  public double d;
  public String s;
}
```

The above code creates a class called "ttest". If you save it as "ttest.java" and compile it using command "javac ttest.java", you will create another file in the same folder ttest.java resides. It is called "ttest.class". Now this class is ready for use. If you were writing another Java program, and your program is within the classpath that leads you to access the ttest class, you can write the following statement to create a Java object:

ttest mytest = new ttest ()

With this statement in place, a new object mytest is created and by using the newly created object, you can access the fields of the class.

Since we are working in the SAS environment, our creation of the java object must happen within the SAS environment. To this end, SAS provides the following example:

```
data _null_;
  dcl javaobj j("ttest");
  length val 8;
  length str $20;
  j.setIntField("i", 100);
  j.setDoubleField("d", 3.14159);
  j.setStringField("s", "abc");
```

```
  j.getIntField("i", val);
  put val=;
  j.getDoubleField("d", val);
  put val=;
  j.getStringField("s", str);
  put str=;
run;
```

Notice the following important points from the above program:

1) dcl javaobj j("ttest"); - creates a Java object
2) j.get*, j.set.* are Java setters and getters methods.
3) The rest of the program are pure SAS.

By writing such a program, we are letting SAS follow Java's rules in order to utilize Java's power which sits outside SAS. One use of this external power is for creating Power Point presentation which represents the effort of integrating SAS with MS office.

3. Creating Power Point Presentation using SAS and Java

To create power point presentation from SAS by following Java's rules, we need to go through 3 steps: to set up necessary Java environment, to create an external Java class and compile it, and to create a Java object from within SAS DATA step by utilizing the compiled Java class.

The first step, setting the necessary Java environment , consists of the following:

1) Download older Java version from Sun site and install it on your machine.
2) Download Apache Poi package from Apache POI side, unzip it to a folder, says, c:\sasjava\lib,
3) Go to start, control panel, system, advanced, environment variables, and in the block of "system variables" create a new environment variable called "CLASSPATH", and gives it the following value:

   .;C:\Program Files\Java\jre6\lib\ext\QTJava.zip; c:\sasjava\lib ;c:\sasjava\lib \poi-ooxml-schemas-3.7-beta3-20100924.jar; c:\sasjava\lib \poi-scratchpad-3.7-beta3-20100924.jar; c:\sasjava\lib \poi-ooxml-3.7-beta3-20100924.jar; c:\sasjava\lib \poi-examples-3.7-beta3-20100924.jar; c:\sasjava\lib \poi-contrib-3.7-beta3-20100924.jar; c:\sasjava\lib \poi-3.7-beta3-20100924.jar;

The second step is to write a java program. Try to create a file called powerPoint.java in c:\sasjava\lib and copy the code from Appendix A and then issue the command "javac powerPoint.java". Now using dir command, you should see a new file called "powerPoint.class" created in c:\sasjava\lib.

The third step is very simple: we can simply copy the SAS sample and do minor changes to satisfy our needs. The following few lines would suffice:

```
data _null_;
 dcl javaobj j("powerPoint");
 j.callVoidMethod("main");
run;
```

The program simply creates a Java object using powerPoint class, and then calls "main" method on the object. If everything goes well, you should see a powerpoint slide show, a ppt file in c:\sasjava\lib. Figure 1 and Figure 2 are results from running the program:

Figure 1

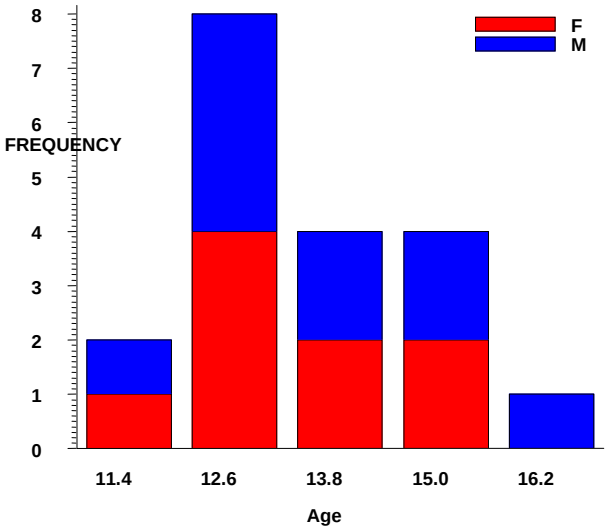

Figure 2

1

Obs	Sex	Total Records	Sample Mean	Standard Deviation	Minimum	Maximum
1	F	9	13.2222	1.39443	11	15
2	M	10	13.4000	1.64655	11	16

4. Limitations of Taking SAS as Part of Java

The previous discussions reveal the power of bringing Java into SAS by creating java objects and communicating with external java programs. However, due to the different language nature of SAS and Java, the power is limited.

One case worth examining is for creating a table in the power point. To do this, we have two options:

1) Do necessary SAS programming to get required statistics and then pass the information to our Java program.

2) In the Java program, we use what SAS has passed and use the to create a table:

Here is a block of code we used:

```
String [][] data = {
        {"INPUT FILE", "NUMBER OF RECORDS"},
        {"Item File", "11,559"},
        {"Vendor File", "300"},
        {"Purchase History File", "10,000"},
        {"Total # of requisitions", "10,200,038"}
    };


   // SlideShow ppt = new SlideShow();

    Slide slide2 = slideShow.createSlide();
    //create a table of 5 rows and 2 columns
    Table table = new Table(5, 2);


    for (int i = 0; i < data.length; i++) {
       for (int j = 0; j < data[i].length; j++) {
          TableCell cell = table.getCell(i, j);
          cell.setText(data[i][j]);
```

```
            RichTextRun rt = cell.getTextRun().getRichTextRuns()[0];
            rt.setFontName("Arial");
            rt.setFontSize(10);

            cell.setVerticalAlignment(TextBox.AnchorMiddle);
            cell.setHorizontalAlignment(TextBox.AlignCenter);
        }
    }
```

As we can see from the above code, the "String [][]" defines a Java data two dimensional array which is required to produce the power point slide. Using set method from within SAS data set, we can only pass a string to our Java program, not a two dimensional array. In order to convert a string into a two dimensional Java array, an additional step is needed:

```
String string = "key1|value1#key2|value2";
String[] first = string.split("#");
String[][] result = new String[first.length][];
for (int i=0; i<first.length; i++)
        result[i] = first[i].split("\\|");
```

Creating table is only one example. Other information needs to be passed from SAS to Java too. In some scenarios, we might need a List data type or a Map data type from within Java program. Therefore, due to incompatibility between the two languages, a smooth passing of information is not easy.

All these difficulties have to do with our initial approach: taking SAS as part of Java by letting SAS mimicking Java's object. Along this line of thinking, that is, by sticking to the principle of "SAS as part of Java", we can do a lot of useful jobs. However, to dynamically run our programs based on data, we'd *better off adjusting our thinking from "SAS as Part of Java" to "Java as Part of SAS".*

5. Java as Part of SAS (JPS)

How different "Java as Part of SAS" from "SAS as Part of Java"?

Well, in fact, JPS builds on SPJ: we still need to create Java objects from within SAS DATA step. The difference only begins when we start to plan where we edit and compile our Java programs.

SPJ edits Java programs independently from SAS: you completely merge yourself as a Java programmer using all the tricks a Java programmer uses, using Eclipse, or Netbeans, for example.

JPS, however, sticks to SAS, even for writing Java programs to run or compile Java programs. Once the Java environment is set up. All you need to do is to open a SAS session, write SAS programs to edit and to run Java and SAS programs.

I must admit that SAS is not good in writing text files compared to many other languages. Still, I can use simple PUT statement to write whatever I want. The SAS program is not pleasant to read, but the result

is not that different from using other languages in terms of the SAS program it produces. With helps of various string functions (see **Ronald Cody, Ed.D, 2007**), we can do almost whatever we want.

The program at saslab.org serves as an example of approach of "Java AS Part of SAS". Here are main highlights of the program:

1) powerPoint.java is written from within SAS program.
2) powerPoint.java is compiled from within SAS program
3) the structure and contents are determined by SAS internal logics.
4) What slides to produce.
5) What is the order of the slides in the slide show.
6) Within the creation of a particular slide, SAS controls the following information passed to the Java program:

   i)Type of the graphs: figure, table, lists

   ii) title for each slide.

   iii) number of slides

   iv) order of slides.

We can see from the program that the communication between Java and SAS is seamless. SAS controls the power point generation process not by passing information via Java methods, but by writing the Java programs. This approach opens endless opportunities of using Java from within SAS: since the text file generation from within SAS is controlled from within SAS its self, the whole Java programming activities is data driven.

Appendix B is just a start for building macros for automatically generating Power Point slides shows. The further enhancement of the program can be along two lines: A) to bring more Java packages for use, for examples, the classes for collecting information from existing slides shows; B) to analyze the slides information collected for further use; C) to re-arrange the slides show so that the short coming of Apache POI package could be overcome by using SAS.

6. Discussions

Generating power point presentation using SAS step Java object is useful only if the computing environment is under Unix. In the window environment, we will be better off if we simply use the approach as discussed by Ya Huang. VBA is much more powerful than Java code as used here because VBA is the natural language for office product.

Generating power point presentation, however, is only one of the functionalities of Office Integration approach as illustrated in the following table:

Table 1 Possible Use of Java as part of SAS

No.#	Job	Explanations
1	Office Integration	PPT, RTF, XML
2	Database Integration	Persistence of SAS data in relational databases (Oracle, PostGreSQL, mySQL)
3	Website Building	Websites
4	Desktop Application	eclipse RCP

As you can see from the above Table, Java nicely complements SAS in the field of presentation. Added to SAS ODS, Office Integration using Java provides more power and more flexibility on other operating systems. Using Jdbc library and Java object, SAS can seamlessly integrate with various relational databases – ODBC can be avoided (and therefore the license fee related to it). Using some popular open sources frameworks (Spring Framework, eclipse RCP), we can use SAS to write SAS code and build our own website and desktop applications dynamically from SAS session. Interested readers can find the updated development in the following link: http://saslab.org/?q=node/55.

7. References

To download all the code that are used in this paper, click the following link:

http://www.saslab.org/?q=node/63

Official Site of Apache POI

http://poi.apache.org/

Sources of Java Code

The Java part of the programs in this paper are adopted from the following two sources:

http://poi.apache.org/slideshow/how-to-shapes.html#NewPresentation

http://www.roseindia.net/java/poi/createPowerPointSlide.shtml

Power Points and VBA

Ya Huang, Amylin Pharmaceuticals, Inc., San Diego, CA, Automate PowerPoint Slide Generation with ODS and VBScript, http://support.sas.com/resources/papers/proceedings10/228-2010.pdf

How to get SAS data into PowerPoint? using - AD04 How to get SAS® DAtA http://corp.comsys.com/docs/analytics/HowtoGetSASDataIntoPPTUsingSAS9.pdf

Aileen L. Yam, PharmaNet, Inc., Princeton, NJ, AUTOMATING THE PRODUCTION OF CUSTOMIZED POWERPOINT PRESENTATION GRAPHS BY INTEGRATING THE FUNCTIONALITY OF SAS® SOFTWARE WITH MICROSOFT VISUAL BASIC FOR APPLICATIONS http://www2.sas.com/proceedings/sugi24/AppDevel/p23-24.pdf

David Bosak, Comsys IT Partners, Kalamazoo, MIHow to get SAS® data into PowerPoint™ using SAS9 http://www2.sas.com/proceedings/sugi30/006-30.pdf

How to deal with Text Files from SAS

Jennifer Lin, A MACRO TOOL TO SEARCH AND REPLACE PORTIONS OF TEXT http://www2.sas.com/proceedings/sugi24/Coders/p086-24.pdf

Ronald Cody, Ed.D, Having a Ball with Strings: SAS Character Functions http://www.nesug.org/proceedings/nesug01/bt/bt3002.pdf

Ronald Cody, Ed.D., An Introduction to SAS® Character Functions http://www2.sas.com/proceedings/forum2007/217-2007.pdf

8. Contact Information

Laiju Zhang, Ph.D

Sr. Manager of Biostatistics & SAS programming/Sr Biostatistician
Medical Device Consultants, Inc.

49 Plain st, North Attleboro, MA 02760
(508) 316 7163

(508) 643-2237 Fax
LZhang@mdci.com
http://www.mdci.com