# A Web-Enabled Reporting Prototype Using JSP, JavaBeans, and SAS ®

**Tim Liu, Health Products Research, Whitehouse NJ**
**Long Zhu, Prudential Financial, Newark NJ**

## ABSTRACT

This poster describes a web-enabled reporting prototype consisting of JavaServer Pages™ (JSP™), JavaBeans™, and SAS tools. This lightweight prototype allows users to initiate SAS processing and retrieve results through the standard browser interface. It shows that all tasks such as verifying login, registering new users, accessing database tables and running SAS can be implemented by JavaServer Pages and JavaBeans interacting with web server, relational databases and SAS.

## INTRODUCTION

One of the strongest capabilities of SAS is its information processing power. On the other hand, web technologies allow for the most effective way of delivery and distribution of information. Combining the best of both worlds, we can build powerful information delivery applications. Advantages of such applications also include standardized user interface to access information sources, reduced client-side software and maintenance, centralized processing and increased security features.

A CGI-SAS approach would serve our purpose. However, it often suffers limitations in areas such as session management and portability.

A better alternative is to replace CGI with a Java-enabled middleware. Stevens and Santucci[1] describes an integrated architecture of Java, JSP, WebLogic™, LDAP and SAS through SAS/Integration Technologies. Here we introduce a streamlined design that features an integration of JSP, JavaBeans, and parameter-driven SAS processing, and does not include the use of SAS/Integration Technologies. It can be viewed as a quick and lightweight solution for web-enabling SAS applications. In addition, by adding a database tier to the design, we demonstrate that access control and user authentication can be implemented through a relational database. As an example, an on-line reporting prototype is developed based on this design.

* * * * * * * * * * * * * * * * * * * * * * * * * * *

## IMPLEMENTATION

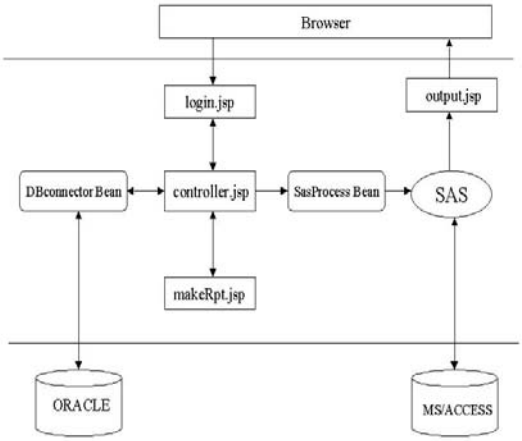



**Figure 1. A simple view of the web-enabled reporting prototype**

A simplified picture of the structure of our prototype is shown in Figure 1. First, the user logins in from the *login* page, and request is forwarded to the *controller* page. In that page, Java Beans are instantiated. For example, a DBConnectorBean is instantiated to establish a connection for accessing an Oracle™ database, where user account information is stored. The *controller* page functions as central moderator communicating among the JSP pages. After successful login, it will pass the user to the *makeRpt* page, where the user can choose how the report can be run. After the user has selected and submitted the report options, the *controller* page instantiates a SasProcessBean. This instance kicks off a SAS job with all the user-input parameters passed in. The SAS job will extract data from a sample database in MS/ACCESS™ and perform joins and computation. After the SAS job is run, HTML output via ODS will be generated and included in the *output* JSP page, which is to be forwarded to upon completion of the SAS job.

In reality, this prototype does have more JSP pages and Java Beans than those shown in Figure 1. For

example, there is a Registration Bean responsible for registering new users by taking in user information from a web page and inserting the record into Oracle database tables.

We would like to elaborate on how this prototype allows the user to create a customized report. Figure 2 shows a web page where the user can choose which sections of the report to run and what time window to report upon. This is realized in the *makeRpt* page, which has been taken over by the *controller* page during the process. After the selections are submitted, the *controller* page will instantiate a SasProcessBean behind the scene.

The part of the *controller* page instantiating SasProcessBean and setting its properties is shown below:

```
<%
    ...

    SasProcessBean SasProcessBeanInstance = new SasProcessBean();

    SasProcessBeanInstance.setUserName(loginBeanInstance.getUserName());

    String[] section_list = request.getParameterValues("sections");
    SasProcessBeanInstance.setSectionList(section_list);

    SasProcessBeanInstance.setBegnMonth(request.getParameter("begnmonth"));
    SasProcessBeanInstance.setEndMonth(request.getParameter("endmonth"));

    SasProcessBeanInstance.setNobsSect1(request.getParameter("nobs1"));
    SasProcessBeanInstance.setNobsSect2(request.getParameter("nobs2"));

    SasProcessBeanInstance.execute();

    ...
%>
```

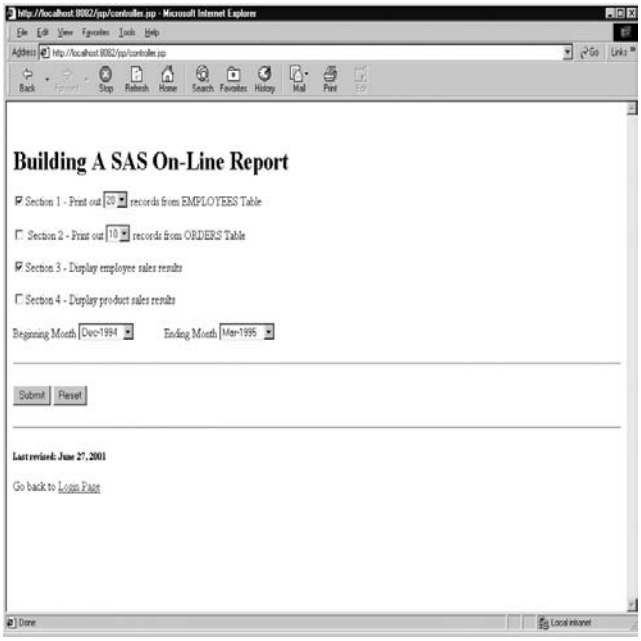
The execute() method of SasProcessBean will run a SAS batch job with all the parameters passed in such as user name, section list, and beginning/ending months. The SAS output is rendered into HTML format via ODS. Then the *output* JSP page includes the ODS output and delivers it to the user.

## CONCLUSION

This prototype is simple and easy to deploy, while it allows users to fully customize the user interface through HTML and JSP programming and implement their reporting requirements through JavaBeans and SAS tools. This prototype can be easily expanded to meet more complicated business rules.

* * * * * * * * * * * * * * * * * * * * * * * * * * * *

**Figure 2. The *makeRpt* page enables the user to choose which sections**




**of report to run and what time window for data inclusion.**

## REFERENCES

[1] Stevens, Jay L, and Brian Santucci, "Integrating SAS with an Open World: Java, JSP, LDAP and Oracle," Proceedings of SUGI 26, Paper 143-26

## ACKNOWLEDGEMENTS

SAS is a registered trademark of the SAS Institute Inc., Cary, NC, USA.

® indicates USA registration. Other brand and product names are trademarks of their respective companies.

## AUTHOR CONTACT

Tim Liu
Health Products Research
3498 Route 22 West
Whitehouse, NJ 08888-0500
e-mail: tim.liu@netzero.net

Long Zhu
Prudential Financial
751 Broad Street, NJ-01-15-07
Newark, NJ 07102
e-mail: long.zhu@prudential.com